DECUS
PROGRAM LIBRARY

DECUS NO.	8-489
TITLE	SUBSET, INTEGER COMPILER AND OPERATING SYSTEM
AUTHOR	R. F. LaFontaine
COMPANY	CSIRO Highett, Victoria, Australia
DATE	September 22, 1971
SOURCE LANGUAGE	MACRO-8

CONTENTS

DECUS Program Library Write-up DECUS NO. 8-489

1. SUMMARY

SUBSET has been developed for the 4000 word PDP8 computers. It comprises a one-pass compiler which interprets FORTRAN/FOCAL-like source programs, and a double precision integer operating system. Its features are the ease which the system can be expanded by relocatable binary subroutines, and a reasonable amount of free memory available for the user's program (5600 octal words).

2. INTRODUCTION

On numerous occasions the programmer is confronted with a project not easily tackled by the use of Fortran or Focal because of the difficulty integrating his subroutines and special input and output devices with these systems. Under these circumstances it is usual to resort to languages such as Pal and Macro8. However, should the program have limited use, the number of programmer-hours may be prohibitive if assembler languages are used.

Subset offers an alternative approach to those problems where double precision integer arithmetic operations can be tolerated. It enables the programmer to construct a library of binary subroutines which are readily coupled with Subset coded calling programs. Subset creates all necessary memory address linkages between program and subroutines, and enables the programmer to reference the subroutines by name in his program.

Subset requires the adoption of the DECUS 8-130A relocating binary loader. Since the storage requirement for this loader is similar to the DEC-08-LBAA-LA loader, and 8-130A loads standard DEC programs, no problems are envisaged.

Programmers with Focal and Fortran experience should not have difficulties writing Subset programs. A similarity to these languages is illustrated in the simple program below. Reference to a two-parameter supplementary binary subroutine ANALOG, which is supplied by the programmer to the operating system prior to program start-up, is also shown.

```
    PROGRAM CHECKCHAN
25  KEY !,"CHANNEL",C
    ANALOG C,X
    TYPE"=",X
    GOTO 25
```

2.1 THE COMPILER

The object of the compiler is to re-organize the source program so that the number of operations required during program execution is kept to a minimum. This reduces the memory requirements of the operating system, and also reduces program execution time.

The duties of the compiler are summarized below.

(A) Assign memory locations for the variables and arrays.

(B) Preset memory locations with constants supplied by the data statement.

(C) Convert command and subroutine names to a numeric code.

(D) Convert decimal constants to binary base constants.

(E) Re-organize subscripted variables, parameters, arithmetic expressions, text strings, and statement labels.

(F) Extract comments.

(G) Sense for source program errors and supply diagnostics.

The compiler outputs the converted program in binary format on paper tape, and the tape is labelled with the program name. Compiler operation and diagnostics are discussed later.

2.2 THE OPERATING SYSTEM

The operating system provides the arithmetic, command and parameter handling routines, and also the basic input and output subroutines. It assists loading of the binary program tape produced by the compiler, and supplementary subroutines supplied by the user.

The operating system interprets the binary program as a set of calls to subroutines supplied either by itself (the commands) or by the user (the supplementary subroutines). Should a non-existant command or subroutine be encountered, the system halts with a diagnostic code displayed in the accumulator indicators. Control commands such as GOTO, IF, CALL, etc., cause a search for a statement label in the program stack. As this is done during program execution, control statements can be constructed employing variables and arithmetic expressions for their parameters. For example ;

```
GOTO NEXT=NEXT+1
```

The system contains several utility routines that are available for use by the supplementary subroutines. One of these is the parameter processor which returns numerical values or variable addresses, depending on the requirement of the calling subroutine. Other utility routines include format controlled binary coded decimal input and output

converters that are easily adapted to drive special I/O devices.

The operating system as supplied functions without use of the interrupt facility. It is possible however, to use interrupt through appropriate system patches which can be included in the supplementary binary subroutines.

The utility routines are described in detail in the section "Preparation of Supplementary Subroutines".

3. SUBSET PROGRAM STRUCTURE

A Subset program consists of a set of labelled and un-labelled statements, each containing a command or subroutine name, and usually a set of parameters. The statement labels, commands, and subroutine names are terminated by the space character, and parameters are separated by commas. All other space characters (other than those appearing in a text string), blank lines, rubout codes, and blank tape, are ignored by the system. Statements are terminated by a carriage return, or by the colon (:) if more than one statement per line is desired.

The first statement in the program must be the program name. The name can consist of any string of characters which are reproduced as a label on the leader of the binary program tape by the compiler. The name must be preceded with the word PROGRAM.

The program is terminated by the sequence; carriage return, line feed, dollar sign, carriage return, line feed.

Strict program formatting is not required, that is it is not essential to place statement labels, commands, etc., in prescribed columns, although if this is done, it will aid program reading.

3.1 PARAMETERS

A parameter can be a constant, simple or subscripted variable, arithmetic expression, or a replacement expression. Text strings and format specifications can usually appear in a parameter set, although these are not strictly considered by the system as parameters.

3.2 CONSTANTS

Constants are integer numbers in the range -8, 388, 607 to +8, 388, 607.

3.3 VARIABLES

A variable name must commence with an alphabetic character.

The remaining characters in the name can be any printable character other than

*=:,-/&()+ and the space character.

Only the first two characters of a variable name are used by the system, thus FOX and FOG are considered to be the same variable (FO).

Typical variable names -

Q
SAM
Z36
P.
D$
M%

Illegal variable names -

GA P
MP*
5X

3.4 SUBSCRIPTED VARIABLES

Subscripts must be integer constants or simple variables.

SAV(14)
K(INC)

Refer to the DIMENSION statement for additional information regarding subscripted variables.

3.5 ARITHMETIC AND REPLACEMENT EXPRESSIONS

An expression can contain constants, simple and subscripted variables, and the following operators;

*	Multiply
/	Divide
+	Add
-	Subtract
&	Logical And
=	Replace

Bracketed expressions are invalid and will cause a diagnostic. Expressions are evaluated left to right regardless of the order of the arithmetic operators in the expression.

All arithmetic operations are performed in a 48-bit accumulator which can contain numbers in the range plus-minus 140,737,488,355,327.

When a multiply operation is performed, the low order 24-bits of the 48-bit accumulator are multiplied by a 24-bit constant or variable, leaving the product in the 48-bit accumulator. If the contents of the accumulator (prior to multiply) is outside the range plus-minus 8,388,607 the product will not be valid.

During a divide operation, the content of the 48-bit accumulator is divided by a 24-bit constant or variable, leaving the quotient in the accumulator and the remainder in the variable (R.). It is possible that the divident and the divisor are such that the quotient is outside the range plus-minus 8,388,607 in which case the result will be in error.

Addition and subtraction operations add or subtract a 24-bit constant or variable to the 48-bit accumulator. Add and subtract errors are limited to accumulator overflow (result exceeding plus-minus 140,737,488,355,327).

Logical AND operations "AND" a 24-bit constant or variable with the low order 24-bits of the accumulator. The result is left in the low order 24 bits of the accumulator, the upper 24 bits of the accumulator remain unchanged.

The replace operator (=) causes the low order 24 bits of the accumulator to be transmitted to the variable specified at the left of the replace operator. Usual forms of the replacement expression are ;

```
M=560

BOX(J)=BOX(J)*40+K/100
```

Note that the arithmetic operations in the last expression are performed in the following order (A is the content of the 48-bit accumulator)

```
A=BOX(J)
A=A*40
A=A+K
A=A/100
BOX(J)=A
```

3.6 DIMENSION STATEMENT

DIMENSION statements can appear anywhere in a program provided that the variables that are being dimensioned have not previously been used in the program. DIMENSION statements must not be labelled. The following table shows the equivalences resulting from the DIMENSION statement -

```
                    DIMENSION C(3), B(4), A(2)

A =  A(1) . . . .  B(-1) . . . .  C(-5)
     A(2) . . . .  B(0)  . . . .  C(-4)
     A(3) . . B =  B(1)  . . . .  C(-3)
     A(4) . . . .  B(2)  . . . .  C(-2)
     A(5) . . . .  B(3)  . . . .  C(-1)
     A(6) . . . .  B(4)  . . . .  C(0)
     A(7) . . . .  B(5)  . . C =  C(1)
     A(8) . . . .  B(6)  . . . .  C(2)
     A(9) . . . .  B(7)  . . . .  C(3)
```

All variables in the DIMENSION statement must appear with a dimension. The statement -

```
DIMENSION M,N
```

will cause a diagnostic.

3.7 DATA STATEMENT

The DATA statement is used to preset variables with signed or unsigned integer constants. The DATA statement must not be labelled. DATA statements can appear anywhere in a program provided that subscripted variables used in the statement have previously been dimensioned. The following are examples of DATA statements;

```
DATA U=4500, MAX(40)=-567, L=5

DATA Z(4)=23=69=7=19=800=-11
```

The last DATA statement will preset array Z as follows ;

```
Z(4)=23
Z(5)=69
Z(6)=7
Z(7)=19
Z(8)=800
Z(9)=-11
```

3.8 COMMENTS

Comments can be inserted in the program by prefixing them with the letter C. At least one space must separate the C and the comment, and the comment must not be labelled. An example of a comment :-

```
C DETERMINE LIMIT FOR Q3
```

3.9 SET COMMAND

SET statements can contain any number of parameters. The

parameters are usually replacement expressions, but test strings and carriage control codes can be included, in which case they will be transmitted to the last used device capable of outputting text. The following is an example of the SET statement.

```
SET J=A+8,N(J)=MIN,B=V*C/K5
```

Note that the subscript (J) will have the value A+8.

3.10 GOTO COMMAND

GOTO N

GOTO causes control to pass to statement N. The parameter N may be a constant, variable, arithmetic expression, or a replacement expression. If N specifies a non-existant statement, control passes to the statement following the GOTO statement. Typical GOTO statements are shown below.

```
GOTO 37
GOTO *END
GOTO A=A+1
```

GOTO X,N1,N2,N3,.....

This form of GOTO passes control to statement N1 if X=1, statement N2 if X=2, statement N3 if X=3, etc. When X is negative or zero or is larger than the number of (N) parameters, or the specified statement is non-existant, control passes to the statement following the GOTO statement. Permissible GOTO statements :-

```
GOTO SORT, 127,*SAM,56,67
GOTO B=B+1,S,A+1,M,400,910,3
```

3.11 IF COMMAND

IF X,A

If X is negative control passes to statement A, otherwise control passes to the statement following the IF statement.

IF X,A,B

If X is negative control passes to statement A, if zero passes to statement B, otherwise control passes to the statement following the IF statement.

IF X,A,B,C

If X is negative control passes to statement A, if zero passes to statement B, if positive passes to statement C.

Should the indicated statement (A, B or C) be non-existant control passes to the statement following the IF statement.

X may be a variable, arithmetic or replacement expression.

A, B, C may be constants, variables statement names, or arithmetic or replacement expressions.

The following are examples of IF.

```
IF A=A-1, 37, 17, 28
IF Q(9), Z+K, *BILL, 904
```

3.12 JUMP COMMAND

JUMP S, I, L

Program control jumps to the statement labelled (S) if (I) is less than or equal to (L). When (I) exceeds (L), control passes to the statement following the JUMP statement.

The parameters may be constants, variables, and arithmetic and replacement expressions.

Program loops may be set up using the JUMP command. A simple loop which deposits zero through an array 20 locations in length, is shown below.

```
      .
      .
      SET K=1
41    SET ARRAY(K)=0
      JUMP 41, K=K+1, 20
      .
      .
```

It is permissible to increment or decrement any parameter as shown below.

```
JUMP 100+M, M=M+1, L=L-3
```

3.13 CALL COMMAND

CALL N

The CALL passes control to the subroutine labelled N. If N specifies a non-existant subroutine, the statement following the CALL statement is executed next.

N may be a constant, variable, statement name, arithmetic expression, or a replacement expression. A CALL statement and a Subroutine are shown below.

```
        .
        .
        .
    CALL * RESET
        .
        .
*RESET SET BIN(5)=0, A=J/200
       TYPE "A=", A, U+BIN(K)
       END *RESET
```

3.14 END COMMAND

END N

The END command is used to terminate a subroutine. N will normally be a constant or a statement name, however a variable, arithmetic expression, or a replacement expression may be used. If N specifies a non-existant subroutine, control passes to the statement following END N. If subroutine N has never been referenced by a CALL statement, the entire program is re-executed. Otherwise, control is passed to the statement following the CALL statement that last referenced the subroutine N.

END

An END command without a parameter causes the computer to halt. When the computer continue key is pressed, program operation resumes at the statement following the end statement.

3.15 TYPE COMMAND

The TYPE command is used to output text strings and the value of constants, variables, arithmetic and replacement expressions, on the Teletype. Format specifications and carriage control codes can also be included in this statement.

The TYPE command defines the Teletype as the current output device and will remain as such until another device capable of outputting alpha-numerics is used. The statement :-

```
TYPE "LIST ITEM ",#1,K=K+1,"=",STACK(K)
```

would produce an output similar to :-

```
LIST ITEM 15=8075
```

3.16 KEY COMMAND

KEY is used to input data from the Teletype keyboard or reader. Text strings and carriage control codes can be included in this statement, and will be transmitted to the teletype. The KEY command defines the Teletype as the current output device which will remain as such until another device capable of outputting text is used.

If the current format specification is #0 then a single character only is inputted and its value is stored in the specified variable. For example ;

```
KEY #0, A, B
```

will input two characters from the keyboard and store them in the variables A and B. If the characters were $ and % the values stored in A and B would be 244 and 245 (octal) respectively (refer Teletype octal code list).

If the current format specification is other than #0, deleted characters are ignored and any non-numeric character other than the minus sign preceding a number is ignored. A non-numeric character following a number will terminate it, and the number will be stored in the specified variable. If the terminating character is the back arrow symbol, the number is ignored and can be re-typed.

The following rule applies to the minus sign preceding a number ;

Odd number of minus signs; number is negative
Even number of minus signs; number is posit

Input numbers must be in the range plus or minus 8,388,607.
The following is an example of the key statement.

```
KEY !,"INPUT VALUES OF VIN", VIN, VIN(2),!," DAY ",DAY
```

3.17 READ COMMAND

The READ command inputs data from the high speed paper tape reader. Text strings and carriage control codes will be transmitted to the current output device if included in the read statement. Note that READ does not specify a current output device as does the KEY command. The following is an example of a READ statement.

```
READ A, J, M(1), M(2), M(3)
```

The statement above reads five numbers from paper tape and stores them in the specified variables. The current format specification is assumed to be other than #0, otherwise five characters only will be read and stored. Numeric input can be assured by specifying a format other than #0 in the READ statement ;

```
READ #1, A, J, M(1), M(2), M(3)
```

Refer to the KEY command for additional information regarding the input of numbers and characters.

3.18 PUNCH COMMAND

The PUNCH command outputs data on the high speed paper tape punch. It defines the punch as the current output device, which will remain as such until another device capable of outputting, alpha-numerics is used. Thus all text strings appearing in SET, READ, GOTO, etc., statements following the PUNCH statement, will be directed to the punch. An example of a PUNCH statement is given below.

```
PUNCH !,"HOURLY TOTALS", #8, D6, D7, A, A(2), A(3), A(4)
```

3.19 TEXT STRINGS

Text strings are identified by enclosing them in double quotes ("). When encountered they are transmitted to the last used output device capable of reproducing text. The statement,

```
SET A=0,"A CLEARED"
```

would cause the message - A CLEARED - to be transmitted to the Teletype if it was the last used device capable of outputting text.

Care should be taken when using text strings in the control commands. The statement ;

```
GOTO "RESTART", 20
```

will output the message RESTART before control is passed to statement 20. The statement ;

```
GOTO 20,"RESTART"
```

will not output the message unless a statement labelled 20 does not exist.

3.20 CARRIAGE CONTROL

A carriage return/linefeed is initiated by the exclamation mark (!) appearing in a parameter set. For example;

```
TYPE !,"RESULT=",X
```

The carriage return/line feed codes are transmitted to the device as specified above in text strings.

3.21 STATEMENT LABELS

Labels can be unsigned integer constants or names commencing

with an alphabetic character. Constants must be in the range 0 to 2047, and names must be prefixed by an asterisk (*). The following labels are valid.

```
3
1069
*CODE
*H94
```

Only the first two characters in a statement name are used by the system, thus *JOB and *JOT are considered the same name, (*JO).

During program compilation, statement names are substituted by negative numbers in the following manner.

Statement name	*SAVE	
Equivalent Octal number	-2301	(S=23, A=01)
Decimal equivalent	-1217	

3.22 FORMAT SPECIFICATION

#N

N represents a constant or a simple or subscripted variable. Subsequent numeric printout of constants, variables, etc., are allocated (N) character spaces for each constant, variable, etc.. When specifying (N), the sign character (- for negative values and a space for positive values) should be included in the count. If (N) specifies more character spaces than necessary, printout is right justified with space character fill.

If (N) specifies less character spaces than is necessary, printout is left justified, but sufficient character space is automatically allocated so that a value is properly represented. If the value printed is positive, the sign character (space) is dropped. The effect of (N) on printout is shown below. The asterisk (*) represents space character fill.

VALUE TO BE PRINTED	FORMAT (N)	PRINTED OUTPUT
0	#1	0
1	#1	1
-1	#1	-1
-1	#2	-1
1	#2	*1
-1	#8	******-1
14076	#7	**14076

The format specification #0 is used to output non-standard characters. This specification causes the lower 12 bits of the 48-bit accumulator to be transmitted to an output device without code conversion. For example, the Teletype can be supplied with the BELL code by the statement ;

```
TYPE #0 , 135

(135 decimal = 207 octal = BELL code)
```

The format specification remains in force until a new format is specified.

The following example of format specifications shows a method by which the contents of a variable, previously scaled up by a factor of 1000, can be printed in proper units (as a floating point number). The divide remainder (R.) is utilized, and the method is suitable only for positive values. Assume the variable V contains the value 356789. The statement ;

```
TYPE  1, V/1000 , "." , R./100 , R./10 , R.
```

would produce ;

```
356.789
```

on the Teletype.

4. SUBSET PROGRAM EXAMPLE

The function of the program shown on page 15 is to record a slowly changing voltage and provide a visual display of voltage versus time, together with a histogram representing the spectrum or distribution of the voltage readings. The display is continuously updated on a storage C. R. T. (Fig. 1), and a hard copy, together with additional information supplied by an incremental plotter (Fig. 2) upon request. The program provides automatic scaling so that axis lengths are constant regardless of the duration of program running time or amplitude of recorded data.

The program illustrates the use of supplementary subroutines (ANALOG, TRACE, SCOPE, PLOT) in a Subset source program. The subroutine ANALOG is called to read a voltage from a single channel A. D. C. This subroutine also contains a system program patch which enables program interrupts, the patch is applied by ANALOG the first time it is called. The purpose of interrupt is to enable a six second clock to increment the variable T., interrupts originating from other sources are ignored. Subroutine TRACE is called to write alpha-numeries on a C. R. T., subroutine SCOPE called to display a dot at specified coordinates on the C. R. O., and PLOT called to draw a line and write alpha-numeries on an incremental plotter.

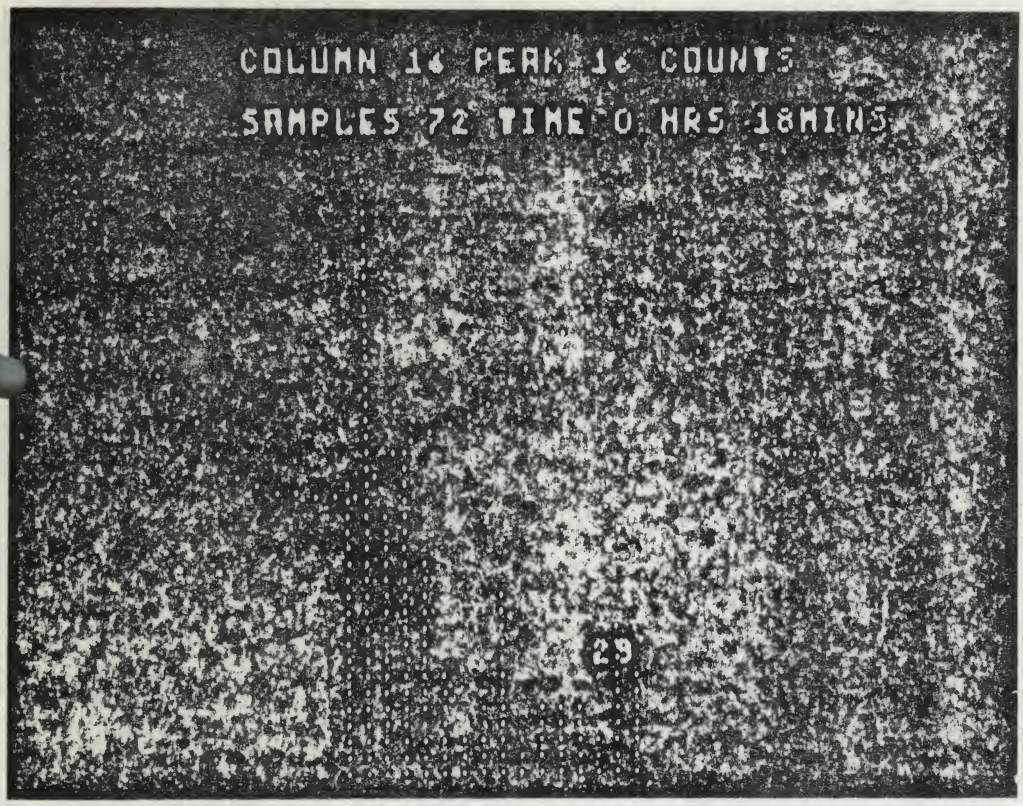
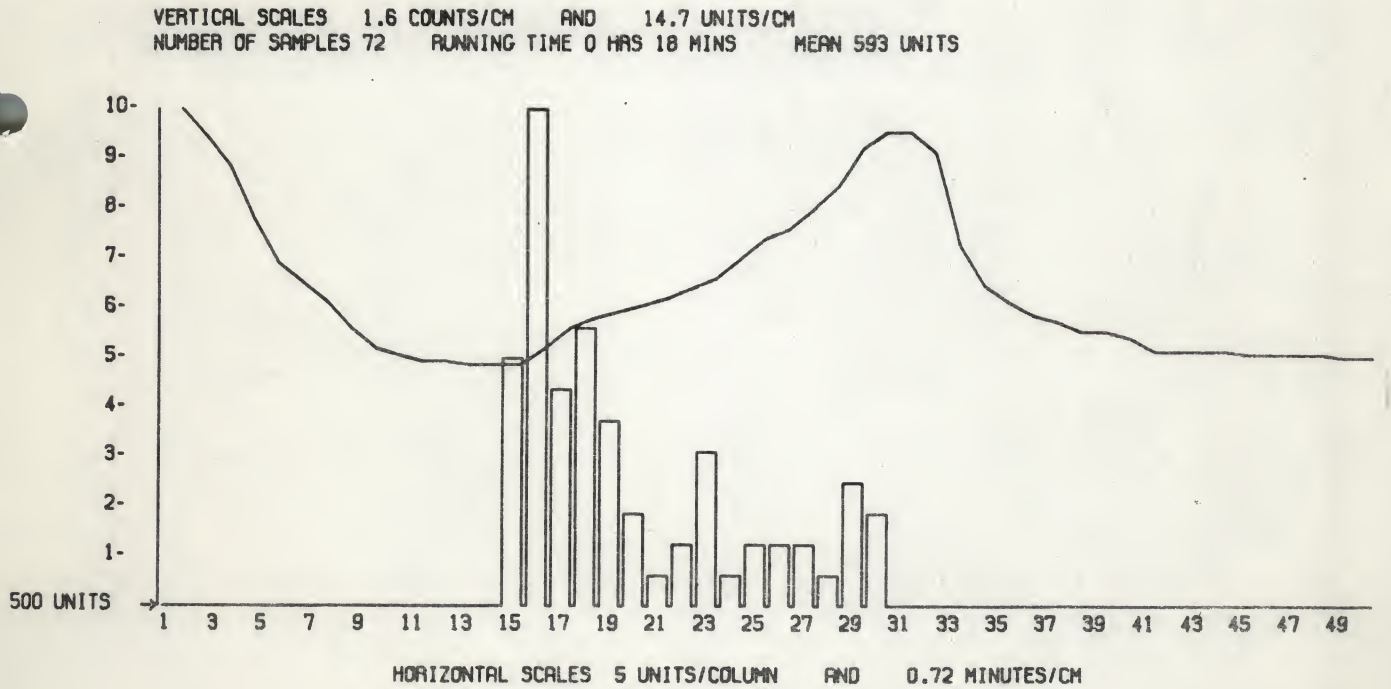


Fig. 1
C. R. T. display.
Last column updated
is indicated (29).



```
        PROGRAM VISILOG
        DATA T.=0,ORIGIN=1,SKIP=2,LINE=3,TEXT=-4,ARROW=-3
        DIMENSION BIN(50),A(50),DISP(50)
        KEY !,"-1 TO START, +1 TO PLOT :",K
        IF K,*START
        CALL *GRAPH
        GOTO *INPUT
*START  KEY !,"CHANNEL ",C," OFFSET ",OFF," UNITS/COLUMN ",S
        SET MAX=1,QMAX=1,K=1,B=0,IT=4,COUNT=1,SAM=0
        SET T.=0,WHEN=4,DV=0,AV=0
   152  SET BIN(K)=0,A(K)=0,DISP(K)=0
        JUMP 152,K=K+1,50
*INPUT  ANALOG C,X
        IF OFF-X,702:SET X=OFF
   702  SET L=X-OFF/S+1,SAM=SAM+1,DV=DV+1,AV=AV+X-OFF
        IF L-51,440:SET L=50
   440  SET BIN(L)=BIN(L)+150,K=1
        TRACE 50,185,#0,0,#1,"COLUMN ",B," PEAK ",MAX," COUNTS"
        TRACE 50,170,"SAMPLES ",SAM," TIME ",T./600," HRS ",R./10,"MI
        SET Z=MAX*150
        IF BIN(L)-Z,507
        SET MAX=BIN(L)/150,B=L
   507  TRACE L*5-6,BIN(L)/MAX,L
        IF T.-WHEN,*SCOPE
        SET TR=T.,A(COUNT)=AV/DV,KK=1,DV=0,AV=0
        IF A(COUNT)-QMAX,1000
        SET QMAX=A(COUNT)
  1000  SET L=KK-1*COUNT/50+1
        SET DISP(KK)=A(L)*150/QMAX
        JUMP 1000,KK=KK+1,50
        JUMP 1004,COUNT=COUNT+1,50
        SET Z=1,QMAX=1,COUNT=26
  1002  SET KK=Z+Z,L=KK-1
        JUMP 1003,A(Z)=A(KK)+A(L)/2,QMAX
        SET QMAX=A(Z)
  1003  JUMP 1002,Z=Z+1,25
        SET IT=IT*2
  1004  SET WHEN=TR+IT
*SCOPE  IF BIN(K),506,505
   506  SET J1=0,POS=K*5,J=BIN(K)/MAX
   600  SCOPE POS,J1=J1+4
        IF J1-J,600
   505  SCOPE K*5+2,DISP(K)
        JUMP *SCOPE,K=K+1,50
        GOTO *INPUT
*GRAPH  SET K=2,TR=T.,Z=DISP
        PLOT ORIGIN,-1000,0,    SKIP,-1018,0,    ARROW,"->"
        PLOT SKIP,-1000,0,      LINE,-1000,1000
        PLOT SKIP,-1300,0,      TEXT,#1,OFF,"UNITS"
        PLOT SKIP,-950,DISP*1000/150
    40  SET Z=DISP(K)+Z
        PLOT LINE,K*50-1000,DISP(K)*1000/150
        JUMP 40,K=K+1,50
        SET K=50
        PLOT SKIP,1500,0
    42  PLOT LINE,X=K*50-1000,0
        PLOT LINE,X,J1=BIN(K)*1000/150/MAX
        PLOT LINE,X-40,J1,      LINE,X-40,0
        IF K=K-1,43,43,42
    43  PLOT SKIP,-1000,1110
        PLOT TEXT,"NUMBER OF SAMPLES ",SAM
        PLOT TEXT,"     RUNNING TIME ",TR/600
        PLOT TEXT," HRS ",R./10," MINS      MEAN "
        PLOT TEXT,Z*QMAX/7500+OFF," UNITS"
        PLOT SKIP,-1000,1160
        PLOT TEXT,"VERTICAL SCALES   ",MAX/10,".",R.
        PLOT TEXT," COUNTS/CM     AND     ",QMAX/10
        PLOT TEXT,".",R.," UNITS/CM"
        PLOT SKIP,-500,-160
        PLOT TEXT,"HORIZONTAL SCALES  ",S
        PLOT TEXT," UNITS/COLUMN     AND     ",TR/250
        PLOT TEXT,".",R./25,R.*2/5," MINUTES/CM"
        SET K=1
    44  PLOT SKIP,K*50-1035,-40,     TEXT,K
        JUMP 44,K=K+2,49
        SET K=0
    45  PLOT SKIP,-110,K=K+100,      TEXT,K/100,"-"
        IF K-1000,45
        PLOT SKIP,-1000,1500
        END *GRAPH
$
```

5. COMPILING SUBSET PROGRAMS

The SUBSET COMPILER must first be loaded into memory using the Binary Loader. At completion of loading, place the Subset program in the Teletype reader, switch the paper tape punch ON, put 0200 in the Switch Register, press Load Address, then press the Start switch. The punch will output leader code and label the paper tape with the program name. Program compilation commences and a binary version of the program is outputted. If the computer halts before program compilation is complete, read the value displayed in the accumulator indicators and check against the list of compiler diagnostics. Press the Continue switch and compilation resumes. If a diagnostic occurs, the resultant binary program tape will be in error and should be discarded. At completion of program compilation, trailer code is punched. The binary program is now ready to be loaded by the SUBSET OPERATING SYSTEM.

6. LOADING SUBSET BINARY PROGRAMS

Load the SUBSET OPERATING SYSTEM into memory using the Binary Loader. It is essential that the REBIL8 binary loader is used. At completion of loading, place the Subset binary program in the reader, put 0200 in the Switch Register, press Load Address, then press Start. The binary program tape will then be loaded, and at completion of loading, the accumulator indicators should be checked for a checksum error. If an error is indicated, it is suggested that the entire binary program loading procedure be repeated. If checksum errors persist, the binary program tape has probably been mis-punched, and it may be necessary to re-compile the program.

NOTE :- The operating system, Subset binary program and supplementary subroutines can be loaded by either the high speed reader or Teletype reader. Only one reader must be ON LINE during loading. Therefore, if the high speed reader is used for loading, the Teletype reader should be placed in the STOP or FREE position or the Teletype switched OFF LINE. Failure to observe this can result in loading errors.

At completion of binary program loading, the program can be started providing that the program does not require supplementary subroutines.

7. PROGRAM START-UP

Put 0202 in the Switch Register, press Load Address, then press Start. The program should commence operation, if not, or if the computer halts abnormally during program operation, read the value displayed in the accumulator indicators and check against the list of execution diagnostics.

8. LOADING SUPPLEMENTARY SUBROUTINES

To load a subroutine, put 0201 in the Switch Register, put the subroutine in the high speed reader, press Load Address, then press Start. At the end of every load, check the accumulator for checksum errors. If an error occurs, do not attempt to reload the subroutine, but repeat the loading sequence commencing at LOADING SUBSET BINARY PROGRAMS. When all the library subroutines required have been loaded, the program can be started at location 202.

The order in which the subroutines are loaded is immaterial so far as program operation is concerned, however it is good practice to load last the subroutine which uses the least number of locations in its last (highest numbered) memory page. The unused locations of the last memory page are then made available for use as arrays and variables.

9. PROGRAM RE-START

The program can be re-started at SA=202, but if program termination is normal, pressing the Continue switch will suffice. If the program terminated on an END statement, execution will resume at the statement following END. If the program terminated at the physical end of program, the entire program is re-initiated. If the program terminated through execution diagnostics, it will be necessary to restart at SA=202.

10. COMPILER DIAGNOSTICS

ACCUMULATOR CONTENTS	DIAGNOSTIC
0363	Statement label incorrectly punched.
0372	Normal program end.
0470	Error to left of equal sign.
0625	A constant incorrectly terminated.
0711	Comma, colon, or carriage return missing.
0650	Illegal code or operation.
0723	Symbol @ misplaced.
1044	Program and variables overlapped.
1204	Error in dimension statement.
1213	Error in dimension statement.
1236	Error in dimension statement.
1246	Error in data statement.
1257	Error in data statement.
1313	Error in data statement.

Note : The content of the accumulator indicates the value of the computer's program counter at time of error detection.

11. EXECUTION DIAGNOSTICS

ACCUMULATOR CONTENTS	DIAGNOSTIC
0234	A non-existant subroutine has been called.
0000 at PC=0256	Physical end of program.
0365	Attempt to deposit in program area.
0372	Attempt to deposit in loader area.
0513	Unknown arithmetic operation, or missing parameter.
0556	Unknown operation. System and/or program probably damaged.
0000 at PC=0707	END encountered. Press Continue to resume operation.

12. PREPARATION OF SUPPLEMENTARY SUBROUTINES

Binary subroutines are produced using the assembler languages. They are then converted to relocatable binary using the DECUS 8-130B (RELCON) binary to relocatable binary converter. Subroutines must fulfil the following requirements;

(1) They must not be assembled in page zero. Refer to the DECUS 8-130B write-up for other restrictions.

(2) Location O of the first page occupied by the subroutine must contain the first two ASC II characters (trimmed to six bits) of the subroutine name. For example, the subroutine named SWITCH is coded 2327 (S=23, W=27). Subroutine names must not have the same leading characters as those of the command names, or of other subroutines that are likely to be loaded at the same time.

(3) Location 1 and 2 of the same page must contain the value zero.

(4) Location 3 of the same page must contain the subroutine's first instruction.

(5) Subroutines should terminate with a JMP I 74 instruction, or an effective JMS 744, JMP 1000 or JMP 1600.

The following subroutine fulfils these requirements. Its function is to input a value from the switch register and store it in the variables prescribed by the calling program.

```
*200
0200    SWITCH,     2372            /SUBROUTINE NAME
0201                0               /USED BY THE OP. SYSTEM
0202                0               /USED BY THE OP. SYSTEM
0203                JMS I 35        /GET ADDRESS OF FIRST VARIABLE
0204                LAS             /INPUT FROM SWITCH REGISTER
0205                DCA I 40        /STORE NUMBER IN VARIABLE
0206                DCA I 37        /STORE NUMBER IN VARIABLE
0207                JMS I . +2      /ANY MORE PARAMETERS?
0210                JMS SWITCH+3    /YES; REPEAT
0211                744             /ADDRESS OF TERMINATING
                                        SUBROUTINE
```

Because of its short length, subroutine SWITCH should be loaded after other supplementary subroutines. This will release 166 octal words (377-212) of memory for use as variables and arrays.

A Subset source program reference to subroutine SWITCH is shown below.

```
SET M=700,N=20
SWITCH V, V(2), V(3)
GOTO 37
```

12.1 RELOCATABLE BINARY TAPE GENERATION

Reference should be made to the Relcon (DECUS 8-130B) relocatable binary converter write-up, but the general procedure for converting Subset binary subroutine tapes, as produced by Macro8 or Pal, to relocatable binary is given here. The following information must be supplied to Relcon before conversion takes place;

(1) The lowest address occupied by the subroutine, as specified in the unconverted binary tape. For the subroutine SWITCH, this address is 200. Note that subroutines can be developed in any memory page other than page zero.

(2) The address of locations that contain addresses used for indirect address referencing within the subroutine. The subroutine SWITCH contains only one address located in 0211 (0744), however this address is external to SWITCH and is therefore not declared. The following represents a section of a subroutine and shows the addresses that must be declared.

```
                        *1000
                        .
                        .
1023    7006            RTL
1024    1307            TAD LIST
1025    3306            DCA TEMP
1026    1706            TAD I TEMP
1027    5705            JMP I NEXPAG
        .               .
        .               .
1105    1200 NEXPAG,    TEST      /DECLARE
1106    0000   TEMP,    0
1107    1110    LIST,   .+1       /DECLARE
1110    7033            7003
1111    5162            5162
        .               .
        .               .
1200    7510    TEST,   SPA
1201    1236            TAD A
        .               .
```

To convert the above binary subroutine to relocatable binary, the following data tape must be prepared;

*1000 1105 1107 .... $

Briefly, the procedure for conversion is as follows;

1. Load Relcon into memory
2. Place the above data list in the tape reader
3. Start Relcon and the list will be read. If reading stops before the entire list is read, a list format error has been detected.
4. Place the binary subroutine in the reader, switch the paper tape punch on, then press the computer continue key.
5. Conversion commences. If trailer code fails to copy, a binary tape reading error has occurred and it is necessary to repeat steps 1 through 5.

12.2 UTILITY REGISTERS

The operating system employs the following registers for the communication of information to and from the supplementary subroutines.

MEMORY LOCATIONS 20, 21, 22, 23

These locations comprise the 48-bit accumulator. Location 20 contains the most significant part, location 23 the least significant. Whenever a parameter is processed, its value is left in these locations. The parameter X=3 will leave the value 3 in location 23, and zeros in locations 20, 21, and 22.

MEMORY LOCATIONS 37, 40

These locations contain the high and low order addresses (respectively) of the last variable operated on by the parameter processor. The parameter X=3+Y will leave the memory addresses occupied by the variable X in locations 37, 40.

MEMORY LOCATIONS 7576, 7577

These locations contain the 24-bit remainder (R.) from the last divide operation. Location 7576 contains the most significant portion, 7577 the least significant.

MEMORY LOCATION 42

This address contains the numeric value of the current format specification.

MEMORY LOCATION 103

The address of the last used alpha-numeric output device subroutine is held in this location.

12.3 UTILITY ROUTINES

12.3.1 SUPPLEMENTARY SUBROUTINE TERMINATION

REFERENCE :- JMS 744 (EFFECTIVE)

REFERENCE :- JMP I 74

This instruction (JMS 744) is used when a supplementary subroutine must handle an unspecified number of parameters. When called, it hands control back to the operating system if the parameter list is exhausted, otherwise control returns to the supplementary subroutine. Its use is shown in the example subroutine SWITCH.

The instruction (JMP I 74) terminates a supplementary subroutine which uses a fixed number of parameters. The subroutine SWITCH could use this instruction in place of the instruction (JMS I .+2) located at 207. In which case, SWITCH would use only one parameter, and the instructions located at 210 and 211 become superfluous.

12.3.2 PARAMETER PROCESSOR

REFERENCE :- JMS I 35

This subroutine is called each time a parameter is to be evaluated. It returns the value of the parameter (the value of an arithmetic or replacement expression, or the value of a constant, or a variable) in the 48-bit accumulator. It also returns the address of the last variable operated on (in locations 37 and 40). (Refer memory locations 37 and 40 in UTILITY REGISTERS). Use of this address is shown in the subroutine SWITCH.

When the parameter processor encounters a carriage control code (!) or a text string, it is transmitted to the last used device capable of outputting alpha-numerics. If a format specification is encountered, the content of location 42 is altered accordingly. The parameter processor does not accept carriage control code, text strings, and format specification as true parameters, and thus proceeds to process the next parameter.

12.3.3 BCD INPUT AND OUTPUT ROUTINES

A supplementary subroutine that references a BCD input or output routine comprises two parts;

(A) A subroutine that is called by name in the Subset program, and which defines the address of the subroutine (B) that services the input or output device. Subroutine (A) has the responsibility of initializing hardware flags and software counters that may be needed by the input/output subroutine (B). The subroutine (A) terminates with an effective JMP 1600 for BCD input operations or JMP 1000 for BCD output operations.

(B) A subroutine that is called by the BCD input/output routine, and has the responsibility of providing code translation as may be necessary between the input/output equipment and the 8-bit ASCII BCD input/output routine. In addition, subroutine (B) must generate a non-numeric ASCII code to terminate a number when an input device is incapable of doing so.

The BCD input/output routine contains the terminating instruction JMS 744 , so that Subset program calls to supplementary subroutines using the BCD input/output routine can contain an unlimited number of parameters. However, the input/output device may only be capable of transmitting a fixed number of characters, and it is therefore the responsibility of the supplementary subroutine call to contain only sufficient number of parameters as can be handled by the input/output device.

12.3.4 BCD INPUT

REFERENCE :- JMP 1600 (EFFECTIVE)

Input devices which are to supply the operating system with BCD

information are programmed in a manner similar to the example subroutine DVM whose function is to input data from a digital voltmeter. This subroutine assumes that the digital voltmeter will supply a code other than 01 to 11 (octal) to signify end of number. The notes on the KEY command apply to DVM, with the exception that an output device is not defined, and the number of parameters contained in a DVM statement is dependent on whether the digital voltmeter is capable of supplying more than one reading.

```
DVM,      0426              /DVM CODE
          0                 /USED BY THE OP. SYSTEM
          0                 /USED BY THE OP. SYSTEM
          IOT XX            /RESET DIGITAL VOLTMETER
          TAD DVADRS        /GET DVM INPUT ROUTINE ADDRESS
          DCA 103           /DEFINE DVM AS INPUT DEVICE
          JMP I BCDINP      /GOTO BCD INPUT ROUTINE
BCDINP,   1600
DVADRS,   DVM2              /DEFINE IN RELCON DATA TAPE
/THIS SUBROUTINE CALLED BY THE BCD INPUT ROUTINE
DVM2,     0
          IOT XXX           /DIGITAL VOLTMETER READY?
          JMP .-1           /NO
          IOT XXXX          /YES; INPUT A BCD CHARACTER
          TAD P260          /ADD 260; NOW IN TELETYPE CODE
          JMP I DVM2        /RETURN TO BCD INPUT ROUTINE
P260,     260
```

A typical call to subroutine DVM is shown in the next example.

```
          .
          .
          SET T=1
       50 DVM#1,IN
          SET A(T)=IN
          JUMP 50,T=T+1,20
          .
          .
```

12.3.5 BCD OUTPUT

REFERENCE :- JMP 1000 (EFFECTIVE)

Output devices capable of handling alpha-numerics are programmed in a manner similar to the TYPE command. An extract from the operating system assembly listing is given below as an example of a typical alpha-numeric output routine.

```
TYPE,     2431              /TYPE CODE
          0                 /USED BY THE OP. SYSTEM
          0                 /USED BY THE OP. SYSTEM
          TAD TELE          /GET TELETYPE OUTPUT ROUTINE ADDRESS
          DCA 44            /DEFINE TELETYPE AS OUTPUT DEVICE
                            /GOTO BCD OUTPUT ROUTINE
          JMP I OPNT
OPNT,     1000
TELE,     TLTYPE            /DEFINE IN RELCON DATA TAPE

/THIS SUBROUTINE CALLED BY THE BCD OUTPUT ROUTINE
TLTYPE,   0
          TSF
          JMP .-1
          TLS
          CLA CLL
          JMP I TLTYPE      /RETURN TO BCD OUTPUT ROUTINE
```

13. BINARY PROGRAM STRUCTURE

On occasions, the diagnostic code displayed in the accumulator is not sufficient to indicate the cause of a program execution error. Additional information may be obtained by examining the content of memory location 0007 when the 0234 diagnostic occurs, as the contents of this location represent the first two characters of the name of the missing supplementary subroutine.

Memory location 0016 is used by the operating system as a program counter and indicates the point in the Subset program which is currently in execution. To make use of the information contained in location 0016, an octal dump around the area of memory specified in this location, and an understanding of the structure of Subset binary programs are necessary.

The binary program consists of binary records which are the counterpart of the source program statements. Binary statement numbers preceed those records whose corresponding source program statements have been labelled. Each record contains the binary coded name of a command or supplementary subroutine (SET command coded 2301; S=23, E=05), and a set of parameters. The parameters comprise a string of operation codes and operands. The operation code specifies the arithmetic operation to be performed, and the nature of the operand (whether a constant, simple variable, variable subscripted with a constant, or a variable subscripted with a simple variable).

The operation code is contained in a 12-bit word and takes the octal form :- 0NXX

where;

N = 0 The operand is a simple variable. The following word contains the variable address.

N = 1 The operand is a variable subscripted by a variable. The following word contains the address of the subscripting variable, the next word contains the address of the first location (element) of the subscripted variable.

N = 2 This operand is a constant. The following two words contain the constant.

N = 3 The operand is a variable subscripted by a constant. The following word contains the subscripting constant. The next word contains the address of the first element of the subscripted variable.

XX=00	(=) REPLACE OPERATION
XX=01	NO OPERATION
XX=02	(*) MULTIPLY OPERATION
XX=03	(+) ADD OPERATION
XX=04	NO OPERATION

XX=05 (-) SUBTRACT OPERATION

XX=06 (&) LOGICAL "AND" OPERATION

XX=07 (/) DIVIDE OPERATION

13.1 CONTROL CODES

The following control codes are also found in the binary program.

0041 Carriage control code (!). This causes a 215 and 212 code to be transmitted to the last device capable of outputting alpha-numerics.

0042
4242 These codes represent (") in the source program and enclose a text string. Text characters are trimmed to six bits and packed two characters per word.

0054 Parameters are separated by this code. It is the counterpart of the comma which separates parameters in the source program.

0077 End of record/start of new record.

0777 End of record/start of new record. The new record is labelled with a statement number.

7777 End of program.

The 0777 code is followed by three 12-bit words that contain a statement number, memory address where the next statement label is found, and a word which is filled with a return address should the statement (record) be called as a subroutine. This word is initially set to contain the address of the start of the program (2002).

The source program statements;

```
      SET K=-1
5     TYPE!, 6,A,M(A)*B(7),"TEXT"
      GOTO .....
```

would appear in an octal dump as follows.

```
0077  2305  0205  0000  0001  0000  7524  0777
0005  4036  2002  2431  0041  0043  0203  0000
0006  0054  0003  7464  0054  0103  7464  7520
0302  0007  6312  0042  2405  3024  4242  0077
0717  ..................
```

This octal listing is interpreted as follows.

```
0077    Start of new record.

2305    Name of command or subroutine is SE(T)

0205    Clear AC (48-bit accumulator) and -
0000                subtract -
0001                the constant 1.

0000    Deposit AC contents in the simple variable.
7524    Located at 7524 (K).

0777    Start of new record.  Record is labelled.
0005    Record labelled 5.
4036    Address where next label found.
2002    Return address.  Not yet filled.

2431    Name of command or subroutine is TY(PE)

0041    Output carriage return/line feed.

0043    Format specification.  Is determined by :-
0203    Clear AC and -
0000                Add -
0006                the constant 6.
0054    End of parameter (format) specification).

0003    Clear AC and add the contents of -
7464    The simple variable located at 7464 (A).
0054    End of parameter.  Output the value of the AC.

0103    Clear AC and add the subscripted variable -
7464    Whose subscript is contained in location 7464 (A) -
7520    and first element located at 7520  (M).

0302    Multiply AC by the subscripted variable -
0007    Whose subscript is 7 and -
6312    First element located at 6312 (B).
0042    End of parameter, output value of AC.  Start of Text.

2405    Output the characters TE (T=42, E=05).
3024    Output the characters XT (X=30, T=24).
4242    End of text string.

0077    Start of new record.  Record unlabelled.

0717    Name of command or subroutine is GO(TO).
```

.
.
.

13.2 ESTIMATING SUBSET PROGRAM SIZE

The following table gives the number of 12-bit computer words used for the various elements of a Subset program.

Statement label	3
Command or subroutine name	1
Variable	2
Reference to a constant	3 (constant included)
Reference to a simple variable	2
Reference to a Subscripted variable	3 (constant included if present)
Carriage control code (!)	1
Comma	1 (end of parameter mark)
Colon	1 (end of statement mark)
Carriage return	1 (end of statement mark)
Arithmetic operator	0 (included with operand)
Data statement	0
Dimension Statement	0
Program name	0
Comment	0
Text	2 Plus 1 word for each two text characters
Format specification	1 Plus 2 words for a simple variable or constant, or 3 words for a subscripted variable specification

14. INTRODUCTION TO THE COMPILER ASSEMBLY LISTING

Invariably, a system is altered to suit individual requirements, or expanded to increase its usefulness. The assembly listing has been included to assist such alterations.

The listing contains comments, and the following summary of the major sections of the program is included so that their function might be more readily understood.

DECOD2

This subroutine sorts constants, simple variable names, text

strings, format specification and carriage control code. When a constant is encountered, control returns to the call+1 with the binary value of the constant stored in memory locations 20 and 21 (most significant part in location 20).

If a variable is encountered, control returns to the Call+2 with the first two ASCII characters (trimmed to six bits) in the accumulator. A single character variable name is coded as though the name commenced with the character zero. For example, the variable (E) is coded 0005 (0E, 0=00, E=05).

Text strings, carriage control (!) and the format code (#) are trimmed to six bits and punched directly on the binary tape.

DECODE

Subroutine DECODE calls DECOD2 to obtain binary constants and variable names from which the binary operand is constructed (refer Binary Program Structure). DECODE calls subroutine LISTER to convert variable names to operating system memory addresses. Part (N) describes the operation code - 0NXX - is formed in location 10. (N) describes the nature of the operand(whether a constant, simple variable, etc.).

EXPBACK

This section of the compiler calls subroutine DECODE to obtain parameter operands, adds the part (XX) of the operation code - 0NXX - and searches for parameter and statement termination. When a replacement expression is encountered, EXPBACK defers punching the replace operation code and its operand till all other arithmetic operation codes and operands have been punched. For example the expression;

A=B*5

is rearranged by EXPBACK to;

+B*5=A

which will cause the operating system to;

(1) Clear AC and add the variable B

(2) Multiply the AC by the constant 5

(3) Deposit the product in the variable A

Note: The operator (+) is assigned to the left most operand if it is unsigned.

MEMORY PAGE 1

This page sorts comments, data and dimension statements,

statement labels, command and supplementary subroutine names. Data and dimension statements are directed to another section of the compiler.

The subroutine DECOD2 is called to obtain statement labels, command and subroutine names. When a statement label is encountered, a binary program patch is punched. This patch represents the address in the operating system where this statement label will be located. The patch is applied to the address following the location containing the last statement label.

LISTER

Subroutine LISTER searches for a variable name in the list of variables already used in the program. If found, LISTER returns the address in the operating system where the variable will reside. Subscripted variables are given the address of the first element of the array.

If the variable name is not contained in the list, the name is placed in the list and an operating system memory address is assigned to the variable.

DIMEN

DIMEN processes the dimension statement. It calls LISTER to obtain an operating system memory address for the first element of the array, then alters the address according to the array's dimension.

DATA

DATA processes the data statement. It calls DECOD2 to obtain variable names, LISTER to determine the variable addresses, and NUMBIN to obtain the data values. The binary data and its operating system memory address is punched on the binary tape.

Memory locations 6600 - 7377

This area of memory contains Decus program LABEL (8-68A) modified to read and punch only the Subset source program name.

Memory locations 1400 - 1555

A slightly modified version of DIGITAL-8-29-U double precision input routine is contained in this area of memory.

15. INTRODUCTION TO THE OPERATING SYSTEM ASSEMBLY LISTING

These notes supplement the comments contained in the assembly listing. Minor changes have been made to the Digital routines used by Subset, therefore the listings of these routines are included.

MEMORY PAGE 1

This section organizes binary program and supplementary subroutine loading. Memory addresses are determined and supplied to the REBIL8 binary loader as address modification (Refer REBIL8 write-up) then control is passed to the REBIL8 loader.

Command and supplementary subroutine names are fetched from the Subset binary stack and control is passed to these routines. The commands, or supplementary subroutines, call subroutine PARAM to obtain either a numerical value or variable address. PARAM is called a sufficient number of times to exhaust the set of parameters in the statement.

PARAM

PARAM directs format specification to FORMAT, carriage control (!) to CRET, and text strings to TEXT0. Parameters comprising constants, variables, arithmetic and replacement expressions, are processed by a call to subroutine EVAL.

EVAL

On entry, EVAL clears the 48-bit accumulator to zero. It then fetches an operation code from the Subset binary program stack and determines the address of the operator routine (add, subtract, divide, etc.) the program is then directed to EVAL3, EVAL5, or EVAL6 depending on the nature of the operand. Simple variables are directed to EVAL4.

EVAL3

Variables subscripted with a variable are directed here. The numeric value of the subscript is obtained and control is directed to EVAL7.

EVAL4

The high and low order addresses of the operand are deposited in the registers ADRESX, ADRESY, and the previously determined operator routine is called to produce a new value in the 48-bit accumulator. Control is then passed to EVAL8.

EVAL5

Constants are directed to EVAL5. The address of the constant is determined and used as the operand address. Control is then passed to EVAL4.

EVAL6

A variable subscripted with a constant is directed here. The value of the constant is obtained and control passed to EVAL7.

EVAL7

The value of the subscript is multiplied by two and added to the address of the first element of the subscripted variable (array) to obtain the operand address. Control is passed to EVAL4.

EVAL8

A test is made to see if the parameter evaluation is complete. If another operation code is detected, EVAL operations are repeated with the exception that the 48-bit accumulator is not cleared to zero.

If parameter evaluation is complete, control exits EVAL, with the value of the parameter in the 48-bit accumulator and the last operand address left in ADRESX, ADRESY.

STFIND

This subroutine is called by the control commands (IF, GOTO, etc.) to determine the address in memory where a specified statement number is located. The search for the statement number commences at the statement number closest the start of the Subset binary program, and continues through the binary program stack. The memory location following the statement number holds the address containing the next statement number. The location following the last statement number in the program contains the value 7777.

If a search for a statement number reaches this point, STFIND causes the remainder of the parameters (in the current statement under process) to be worked off as though parameters of a SET statement.

GOTO

This command checks for the number of parameters. If one parameter only is found, its value is used as a statement number reference, and STFIND is called to determine the address of the statement labelled with this number. Control is passed to this statement.

If more than one parameter is found, the value of the first parameter specifies which of the remaining parameters represents the statement number (refer "GOTO command").

CALL

CALL calls PARAM and STFIND to determine the address of the first statement of a Subset subroutine (source program subroutine). The address of the statement following the CALL statement is saved in the location as noted in "BINARY PROGRAM STRUCTURE".

END

END checks for the number of parameters contained in the

END statement. If no parameters are found, the system halts, waits for the CONTINUE key to be pressed, then control is passed to the statement following the END statement.

If a parameter is found, STFIND is called to determine the address of the statement labelled with a number identical with the parameter value. Control is passed to the point specified by the contents of (this address +2).

IF

The value of the first parameter in the IF statement determines which of the remaining parameters represents a statement number. When the IF statement contains four parameters and the first has a positive value, all parameters are evaluated but only the fourth parameter's value is used to specify a statement number. Execution time is at a minimum when the value of the first parameter is negative, regardless of the number of parameters.

16. DIGITAL & DECUS ROUTINES

These routines are contained in or used by either the compiler or operating system.

Digital 8-14-5
Double precision divide routine

Digital 8-13-F
Double precision multiply routine

Digital 8-25-U
Decimal print routine

Digital 8-29-U
Double precision decimal-to-binary conversion and input

Decus 8-130A
Rebil8 relocating binary loader

DECUS 8-68A
Label

17. COMPILER ASSEMBLY LISTING

```
ADRPCH 1124
ALPHTE 0720
BACKSE 0055
BOT    0114
CHKSUM 0070
CONEND 0246
CONNUM 0324
CON1   0122
CON2   0307
CON3   0317
CRLF   0650
DATA   1240
DATADR 1324
DATA0  0135
DATA2  1315
DATA3  1264
DATA4  1313
DDCB   1400
DECODE 0527
DECOD2 0600
DEC11  0560
DEC12  0555
DEC13  0551
DEC21  0763
DEC22  0733
DEC50  0747
DIMEN  1200
DIMENO 0136
DIM3   1236
ENDCON 0364
ENDPRO 2000
ERROR  0104
EXPBAC 0400
EXPRES 0400
FLAG   0516
FORMAT 0653
INCHAR 0030
INCHAS 0046
K1     0053
K2     0054
LABEL  0137
LASTNU 0072
LFLAG  0130
LINCNT 0103
LISTA  0715
LISTAD 0767
LISTER 1014
LISTES 1000
LISTSY 1061
```

LIS1	1013
LIS2	1057
LIS3	1060
LIS4	1033
LIS5	1021
LTCODE	1130
M10	0134
M11	0764
M162	0120
M2	0101
M200	0100
M21	1327
M212	0113
M25	1325
M3	0064
M32	0133
M34	0115
M376	0124
M40	0063
M42	0132
M43	0770
M5	0123
M50	0565
M51	0045
M52	0131
M54	0520
M55	0524
M6	0766
M72	0126
M75	0526
M77	0066
NUMBIN	1151
OPCODE	0517
OPTEST	0626
PRGLST	0071
PRGSAV	1323
PRSTRT	0117
PUNCH	1141
P100	0076
P16	1237
P177	1150
P2	0073
P200	0077
P2002	0127
P3	0074
P33	0125
P377	0116
P40	0062
P4242	0717
P5	0525

```
P60      0765
P7       0075
P77      0065
P777     0121
READ     0222
REPLAC   0506
SAVNUM   0102
SETEM1   0521
SETEM2   0522
SETEM3   0523
SETEND   0457
SET1     0415
SET2     0453
SET3     0434
SET77    0477
STACK    1075
STACK2   1102
STK1     1146
STK2     1147
TEXT0    0656
TEXT1    0660
TEXT2    0712
TEXT3    0716
TEXT4    0700
VARLST   0067
```

```
                /SUBSET COMPILER PAGE 0

                *30
0030  0000      INCHAR, 0
0031  1416       TAD I 16
0032  1063       TAD M40
0033  7450       SNA
0034  5031       JMP INCHAR+1
0035  1062       TAD P40
0036  3007       DCA 7
0037  1007       TAD 7
0040  1045       TAD M51
0041  7650       SNA CLA
0042  5031       JMP INCHAR+1
0043  1007       TAD 7
0044  5430       JMP I INCHAR
0045  7727      M51, -51
0046  0000      INCHAS, 0
0047  1416       TAD I 16
0050  3007       DCA 7
0051  1007       TAD 7
0052  5446       JMP I INCHAS
0053  5036      K1, JMP INCHAR+6
0054  1063       K2, TAD M40
0055  0000      BACKSET, 0
0056  7240       STA
0057  1016       TAD 16
0060  3016       DCA 16
0061  5455       JMP I BACKSET
0062  0040      P40, 40
0063  7740      M40, -40
0064  7775      M3, -3
0065  0077      P77, 77
0066  7701      M77, -77
0067  0000      VARLST, 0
0070  0000      CHKSUM, 0
0071  0000      PRGLST, 0
0072  0000      LASTNUM, 0
0073  0002      P2, 2
0074  0003      P3, 3
0075  0007      P7, 7
0076  0100      P100, 100
0077  0200      P200, 200
```

```
0100  7600  M200, -200
0101  7776  M2, -2
0102  0000  SAVNUM, 0
0103  0000  LINCNT, 0

0104  0000  ERROR, 0
0105  7200   CLA
0106  1104   TAD ERROR
0107  7402   HLT
0110  7200   CLA
0111  5512   JMP I .+1
0112  0222   READ
0113  7566  M212, -212
0114  7400  BOT, 7400
0115  7744  M34, -34
0116  0377  P377, 377
0117  2000  PRSTRT, 2000        /ADDRESS IN SUBSET OP-SYSTEM
0120  7616  M162, -162
0121  0777  P777, 777
0122  7576  CON1, 7576
0123  7773  M5, -5
0124  7402  M376, -376
0125  0033  P33, 33
0126  7706  M72, -72
0127  2002  P2002, 2002
0130  0000  LFLAG, 0
0131  7726  M52, -52
0132  7736  M42, -42
0133  7746  M32, -32
0134  7770  M10, -10
0135  1240  DATA0, DATA
0136  1200  DIMEN0, DIMEN
0137  6600  LABEL, 6600
0140  0201   201
```

```
                /SUBSET COMPILER PAGE 1
                PAGE 1

0200  5537    JMP I LABEL          /PUNCH LEADER AND LABEL
0201  1122    TAD CON1            /AND INITIALIZE POINTERS
0202  3067    DCA VARLST
0203  3777    DCA ENDPROG+3
0204  3070    DCA CHKSUM
0205  3102    DCA SAVNUM
0206  3103    DCA LINCNT
0207  3071    DCA PRGLST
0210  7001    IAC
0211  3072    DCA LASTNUM
0212  1071    TAD PRGLST
0213  4776    JMS ADRPCH           /PUNCH STARTING ADDRESS
0214  7130    STL RAR
0215  4775    JMS STACK            /PUNCH STATEMENT NO. 4000
0216  7240    STA
0217  4775    JMS STACK            /PUNCH LOC. OF NEXT STAT. NO.
0220  7240    STA
0221  4775    JMS STACK            /PUNCH RETURN ADDRESS
0222  1114   READ, TAD BOT         /READER ROUTINE
0223  3016    DCA 16
0224  6031    KSF
0225  5224    JMP .-1
0226  6036    KRB
0227  7450    SNA
0230  5224    JMP READ+2           /IGNORE BLANKS
0231  1113    TAD M212
0232  7450    SNA
0233  5224    JMP READ+2           /IGNORE LINE FEEDS
0234  1064    TAD M3
0235  7450    SNA
0236  5246    JMP CONEND           /FOUND CARRIAGE RETURN
0237  1120    TAD M162
0240  7450    SNA
0241  5224    JMP READ+2           /IGNORE RUBOUT
0242  1116    TAD P377
0243  0065    AND P77
0244  3416    DCA I 16             /STORE IN BUFFER
0245  5224    JMP READ+2
0246  1065   CONEND, TAD P77
0247  3416    DCA I 16           /USE 77 TO REPRESENT END OF RECORD
0250  1114    TAD BOT
0251  3016    DCA 16               /RESET BUFFER ADDRESS
0252  1054    TAD K2
0253  3032    DCA INCHAR+2
0254  4030    JMS INCHAR           /CALL BUFFER READING ROUTINE
```

```
0255  1066    TAD M77
0256  7450    SNA
0257  5222    JMP READ                /IGNORE BLANK RECORDS
0260  2103    ISZ LINCNT
0261  1125    TAD P33
0262  7650    SNA CLA
0263  5364    JMP ENDCON              /FOUND $ END OF PROGRAM
0264  1053    TAD K1
0265  3032    DCA INCHAR+2            /PERMIT INPUT OF SPACES
0266  4055    JMS BACKSET             /DECREMENT BUFFER ADDRESS
0267  4774    JMS DECOD2              /READ IN A CONSTANT OR VARIABLE
0270  5324    JMP CONNUM              /GOT A CONSTANT
0271  3010    DCA 10                  /GOT A VARIABLE, USE AS INSTRUCTION
0272  1010    TAD 10
0273  1064    TAD M3
0274  7450    SNA
0275  5222    JMP READ                /COMMENT
0276  1124    TAD M376
0277  7450    SNA
0300  5535    JMP I DATA0
0301  1134    TAD M10
0302  7650    SNA CLA
0303  5536    JMP I DIMEN0            /FOUND DIMENSION STATEMENT
0304  1065    TAD P77
0305  4775    JMS STACK               /PUNCH 77 FOR UNLABELLED STATEMENT
0306  1010    TAD 10
0307  4775  CON2, JMS STACK           /PUNCH STATEMENT TYPE
0310  1054    TAD K2
0311  3032    DCA INCHAR+2            /IGNORE SPACES FROM BUFFER
0312  1007    TAD 7                   /FETCH LAST CHARACTER INPUTTED
0313  1066    TAD M77
0314  7650    SNA CLA
0315  5222    JMP READ                /FOUND END OF RECORD
0316  5772    JMP EXPBACK             /EVALUATE AND PUNCH REST OF BUFFER
0317  1007  CON3, TAD 7
0320  1126    TAD M72
0321  7650    SNA CLA                 /LAS CHARACTER  :  ?
0322  5252    JMP CONEND+4            /YES
0323  5222    JMP READ                /READ MORE PROGRAM
0324  1121  CONNUM, TAD P777
0325  4775    JMS STACK               /PUNCH 777 FOR LABLED STATEMENT
0326  1021    TAD 21
0327  4775    JMS STACK               /PUNCH STATEMENT NUMBER
0330  1021    TAD 21
0331  3102    DCA SAVNUM              /SAVE NUMBER FOR DIAGNOSTICS
0332  3103    DCA LINCNT              /SAVE LINE NUMBER FOR DIAGNOSTICS
0333  7240    STA
```

```
0334  4775    JMS STACK          /PUNCH LOC. OF NEXT STATEMENT NO.
0335  1127    TAD P2002
0336  4775    JMS STACK          /PUNCH RETURN ADDRESS
0337  1072    TAD LASTNUM
0340  4776    JMS ADRPCH
0341  1071    TAD PRGLST
0342  1101    TAD M2
0343  3072    DCA LASTNUM
0344  7240    STA
0345  1072    TAD LASTNUM
0346  4775    JMS STACK
0347  7240    STA
0350  1071    TAD PRGLST
0351  3071    DCA PRGLST
0352  1071    TAD PRGLST
0353  4776    JMS ADRPCH         /RESET ADDRESS
0354  1416    TAD I 16           /FETCH NEXT CHARACTER FROM BUFFER
0355  1063    TAD M40
0356  7650    SNA CLA
0357  5354    JMP .-3            /IGNORE SPACES
0360  4055    JMS BACKSET        /DECREMENT BUFFER ADDRESS
0361  4774    JMS DECOD2         /FETCH INSTRUCTION TYPE
0362  4104    JMS ERROR          /GOT A CONSTANT
0363  5307    JMP CON2
0364  7240  ENDCON, STA
0365  4775    JMS STACK          /PUNCH PROG. END CODE
0366  1070    TAD CHKSUM
0367  4775    JMS STACK          /PUNCH TAPE CHECKSUM
0370  4773    JMS LTCODE         /PUNCH TRAILER CODE
0371  4104    JMS ERROR          /PROGRAM END.


0372  0400
0373  1130
0374  0600
0375  1075
0376  1124
0377  2003
```

```
                /SUBSET COMPILER PAGE 2

                PAGE 2
0400   3316     EXPBACK, DCA FLAG
0401   4030      JMS INCHAR          /FETCH CHARACTER
0402   1324      TAD M55
0403   7450      SNA
0404   5210      JMP .+4             /FOUND -
0405   1073      TAD P2
0406   7640      SZA CLA
0407   4055      JMS BACKSET         /NOT +
0410   1007      TAD 7               /GET LAST CHARACTER FROM BUFFER
0411   1324      TAD M55
0412   7650      SNA CLA             /WAS IT -
0413   1073      TAD P2              /YES
0414   1074      TAD P3              /NO; ASSUME +
0415   3317     SET1, DCA OPCODE
0416   4327      JMS DECODE          /GET VARIABLE OR CONSTANT
0417   7200      CLA
0420   1007      TAD 7               /GET TERMINATOR
0421   1326      TAD M75
0422   7650      SNA CLA
0423   5306      JMP REPLACE         /FOUND =
0424   1010      TAD 10
0425   7450      SNA
0426   5253      JMP SET2            /FOUND SIMPLE VARIABLE
0427   1317      TAD OPCODE          /FOUND SUBSCRIPTED VAR. OR CONSTANT
0430   4777      JMS STACK           /PUNCH IT
0431   1011      TAD 11
0432   4777      JMS STACK
0433   1012      TAD 12
0434   4777     SET3, JMS STACK
0435   1007      TAD 7               /GET TERMINATOR
0436   1066      TAD M77
0437   7450      SNA
0440   5257      JMP SETEND           /FOUND END OF RECORD (77)
0441   1325      TAD P5
0442   7650      SNA CLA
0443   5257      JMP SETEND           /FOUND :
0444   1007      TAD 7
0445   1320      TAD M54
0446   7650      SNA CLA
0447   5257      JMP SETEND           /FOUND ,
0
```

```
0450  1007   TAD 7
0451  0075   AND P7                 /DETERMINE OPERATOR
0452  5215   JMP SET1
0453  1317  SET2, TAD OPCODE
0454  4777   JMS STACK
0455  1011   TAD 11
0456  5234   JMP SET3
0457  1316  SETEND, TAD FLAG
0460  7650   SNA CLA                /FIRST OPERATOR = ?
0461  5277   JMP SET77              /NO
0462  1321   TAD SETEM1
0463  4777   JMS STACK              /PUNCH OPERATOR
0464  1321   TAD SETEM1
0465  1100   TAD M200
0466  7650   SNA CLA                /WAS IT A CONSTANT ?
0467  4104   JMS ERROR              /YES
0470  1322   TAD SETEM2
0471  4777   JMS STACK
0472  1321   TAD SETEM1
0473  7650   SNA CLA                /WAS IT A SIMPLE VARIABLE ?
0474  5277   JMP SET77              /YES
0475  1323   TAD SETEM3
0476  4777   JMS STACK
0477  1007  SET77, TAD 7           /GET LAST CHARACTER
0500  1320   TAD M54
0501  7640   SZA CLA                /WAS IT , ?
0502  5776   JMP CON3               /NO; : OR 77
0503  1007   TAD 7
0504  4777   JMS STACK
0505  5200   JMP EXPBACK
0506  1010  REPLACE, TAD 10         /STORE OPERATOR
0507  3321   DCA SETEM1
0510  1011   TAD 11
0511  3322   DCA SETEM2             /STORE VARIABLE OR CONSTANT
0512  1012   TAD 12
0513  3323   DCA SETEM3
0514  7240   STA
0515  5200   JMP EXPBACK
0516  0000  FLAG, 0
0517  0000  OPCODE, 0
0520  7724  M54, -54
0521  0000  SETEM1, 0
0522  0000  SETEM2, 0
0523  0000  SETEM3, 0
0524  7723  M55, -55
0525  0005  P5, 5
0526  7703  M75, -75
0527  0000  DECODE, 0
```

```
0530  4775    JMS DECOD2          /GET VARIABLE OR CONSTANT
0531  5360    JMP DEC11           /GOT CONSTANT
0532  4774    JMS LISTER          /GOT VARIABLE. FIND ADDRESS
0533  3011    DCA 11
0534  1011    TAD 11
0535  3012    DCA 12
0536  1007    TAD 7               /GET TERMINATOR
0537  1365    TAD M50
0540  7640    SZA CLA
0541  5355    JMP DEC12           /SIMPLE VARIABLE
0542  1076    TAD P100            /WAS ( SUBSCRIPED VARIABLE
0543  3010    DCA 10
0544  4775    JMS DECOD2          /GET SUBSCRIPT
0545  5351    JMP DEC13           /FOUND A CONSTANT
0546  4774    JMS LISTER          /FOUND A VARIABLE. FIND ADDRESS
0547  3011    DCA 11
0550  5356    JMP DEC12+1
0551  1021  DEC13, TAD 21
0552  3011    DCA 11
0553  1010    TAD 10
0554  1077    TAD P200
0555  3010  DEC12, DCA 10
0556  1010    TAD 10
0557  5727    JMP I DECODE
0560  1020  DEC11, TAD 20
0561  3011    DCA 11
0562  1021    TAD 21
0563  3012    DCA 12
0564  5354    JMP DEC12-1
0565  7730  M50, -50

0574  1014
0575  0600
0576  0317
0577  1075
```

```
                /SUBSET COMPILER PAGE 3
                PAGE 3

0600  0000  DECOD2, 0
0601  1367   TAD LISTAD
0602  3017   DCA 17                 /STORE LOC. OF VALID TERMINATORS
0603  3130   DCA LFLAG
0604  4030   JMS INCHAR             /GET CHARACTER FROM BUFFER
0605  1131   TAD M52
0606  7440   SZA
0607  5212   JMP .+3
0610  2130   ISZ LFLAG              /FOUND STATEMENT NAME
0611  5204   JMP .-5
0612  1366   TAD M6
0613  7510   SPA
0614  5320   JMP ALPHTEST           /ALPHABETIC ?
0615  1364   TAD M11
0616  7740   SLE CLA
0617  5226   JMP OPTEST             /OPERATOR ?
0620  4055   JMS BACKSET            /FOUND NUMBER
0621  4777   JMS NUMBIN             /CONVERT NUMBER TO BINARY
0622  1007   TAD 7                  /GET TERMINATOR AND TEST
0623  4776   JMS LISTEST
0624  4104   JMS ERROR              /NOT VALID TERM.
0625  5600   JMP I DECOD2
0626  1370  OPTEST, TAD M43
0627  1007   TAD 7
0630  7450   SNA
0631  5253   JMP FORMAT             /FORMAT CODE  #
0632  7001   IAC
0633  7450   SNA
0634  5256   JMP TEXTO               /TEXT CODE "
0635  7001   IAC
0636  7650   SNA CLA
0637  5250   JMP CRLF                /CARRIAGE RETURN CODE  !
0640  1007   TAD 7
0641  1126   TAD M72
0642  7450   SNA
0643  5775   JMP CON3+1              /FOUND :
0644  1123   TAD M5
0645  7450   SNA
0646  5775   JMP CON3+1              /FOUND (77)- END OF RECORD
0647  4104   JMS ERROR
0650  1007  CRLF, TAD 7
0651  4774   JMS STACK
0652  5300   JMP TEXT4
```

```
0653  1007  FORMAT, TAD 7
0654  4774   JMS STACK                 /PUNCH #
0655  5201   JMP DECOD2+1
0656  1007  TEXT0, TAD 7
0657  4774   JMS STACK                 /PUNCH "
0660  4046  TEXT1, JMS INCHAS
0661  1132   TAD M42
0662  7650   SNA CLA
0663  5312   JMP TEXT2                 /END OF STRING
0664  1007   TAD 7
0665  7106   CLL RTL
0666  7006   RTL
0667  7006   RTL
0670  3316   DCA TEXT3
0671  4046   JMS INCHAS
0672  1316   TAD TEXT3
0673  4774   JMS STACK
0674  1007   TAD 7
0675  1132   TAD M42
0676  7640   SZA CLA
0677  5260   JMP TEXT1                 /INPUT MORE TEXT
0700  1315  TEXT4, TAD  LISTA
0701  3017   DCA 17
0702  4046   JMS INCHAS
0703  1773   TAD M54
0704  7650   SNA CLA
0705  5772   JMP EXPBACK
0706  1007   TAD 7
0707  4776   JMS LISTEST               /TERMINATOR (77) : , FOLLOW "
0710  4104   JMS ERROR                 /NO
0711  5771   JMP SET77                 /YES
0712  1317  TEXT2, TAD P4242
0713  4774   JMS STACK                 /PUNCH ""
0714  5300   JMP TEXT4
0715  1070  LISTA, LISTSYM+7
0716  0000  TEXT3, 0
0717  4242  P4242, 4242
0720  1365  ALPHTEST, TAD P60
0721  7450   SNA
0722  4104   JMS ERROR                 /FOUND @
0723  1133   TAD M32
0724  7740   SLE CLA
0725  5226   JMP OPTEST
0726  1007   TAD 7
0727  3363   DCA DEC21
0730  4030   JMS INCHAR
0731  4776   JMS LISTEST               /TERMINATOR ?
0732  5347   JMP DEC50
```

```
0733  1130  DEC22, TAD LFLAG
0734  7640   SZA CLA
0735  5341   JMP .+4
0736  1363   TAD DEC21
0737  2200   ISZ DECOD2
0740  5600   JMP I DECOD2
0741  7240   STA
0742  3020   DCA 20
0743  1363   TAD DEC21
0744  7041   CIA
0745  3021   DCA 21
0746  5600   JMP I DECOD2
0747  1363  DEC50,  TAD DEC21
0750  7106   CLL RTL
0751  7006   RTL
0752  7006   RTL
0753  1007   TAD 7
0754  3363   DCA DEC21
0755  1367   TAD LISTAD
0756  3017   DCA 17
0757  4030   JMS INCHAR
0760  4776   JMS LISTEST
0761  5355   JMP .-4              /IGNORE FOLLOWING CHARACTERS TILL TERM.
0762  5333   JMP DEC22            /GOT TERMINATOR
0763  0000  DEC21, 0
0764  7767  M11, -11
0765  0060  P60, 60
0766  7772  M6, -6
0767  1060  LISTAD,  LISTSYM-1
0770  7735  M43, -43

0771  0477
0772  0400
0773  0520
0774  1075
0775  0320
0776  1000
0777  1151
```

```
          /SUBSET COMPILER PAGE 4
          PAGE 4

1000  0000  LISTEST, 0
1001  7041   CIA
1002  3213   DCA LIS1
1003  1417   TAD I 17
1004  7450   SNA
1005  5600   JMP I LISTEST          /CHARACTER NOT IN TABLE
1006  1213   TAD LIS1
1007  7640   SZA CLA
1010  5203   JMP .-5                /COMPARE NEXT CODE
1011  2200   ISZ LISTEST            /FOUND CODE
1012  5600   JMP I LISTEST
1013  0000  LIS1, 0

1014  0000  LISTER, 0
1015  7041   CIA
1016  3257   DCA LIS2
1017  1260   TAD LIS3
1020  3017   DCA 17
1021  1417  LIS5, TAD I 17          /GET VARIABLE NAME FROM LIST
1022  7450   SNA
1023  5233   JMP LIS4               /END OF LIST
1024  1257   TAD LIS2
1025  7650   SNA CLA
1026  5231   JMP .+3                /FOUND NAME
1027  2017   ISZ 17
1030  5221   JMP LIS5
1031  1417   TAD I 17               /GET ADDRESS
1032  5614   JMP I LISTER
1033  1067  LIS4, TAD VARLST
1034  1101   TAD M2
1035  3067   DCA VARLST             /DETERMINE ADDRESS
1036  1071   TAD PRGLST
1037  1117   TAD PRSTRT
1040  7141   CIA CLL
1041  1067   TAD VARLST
1042  7620   SNL CLA                /PROGRAM AND VARIABLES OVERLAPPED?
1043  4104   JMS ERROR              /YES
1044  1067   TAD VARLST             /NO
1045  3417   DCA I 17               /PUT ADDRESS IN LIST
1046  3417   DCA I 17               /PUT NEW END MARK IN LIST
1047  1017   TAD 17
1050  1064   TAD M3
1051  3017   DCA 17
1052  1257   TAD LIS2
1053  7041   CIA
1054  3417   DCA I 17                /PUT VARIABLE NAME IN LIST
1055  1067   TAD VARLST
1056  5614   JMP I LISTER            /EXIT WITH ADDRESS
```

```
1057  0000  LIS2, 0
1060  2000  LIS3, ENDPROG
1061  0040  LISTSYM, 40              /SPACE
1062  0075   75                     /=
1063  0053   53                     /+
1064  0055   55                     /-
1065  0052   52                     /*
1066  0050   50                     /(
1067  0057   57                     //
1070  0046   46                     /&
1071  0072   72                     /:
1072  0054   54                     /,
1073  0077   77                     /END OF RECORD
1074  0000   00                     /END OF LIST

1075  0000  STACK, 0
1076  7100   CLL
1077  4302   JMS STACK2
1100  2071   ISZ PRGLST             /INCREMENT PROGRAM ADDRESS
1101  5675   JMP I STACK

1102  0000  STACK2, 0
1103  3346   DCA STK1
1104  1346   TAD STK1
1105  7012   RTR
1106  7012   RTR
1107  7012   RTR
1110  0350   AND P177
1111  3347   DCA STK2
1112  1347   TAD STK2
1113  4341   JMS PUNCH
1114  7200   CLA
1115  1346   TAD STK1
1116  0065   AND P77
1117  4341   JMS PUNCH
1120  1070   TAD CHKSUM
1121  1347   TAD STK2
1122  3070   DCA CHKSUM
1123  5702   JMP I STACK2

1124  0000  ADRPCH, 0
1125  7120   STL
1126  4302   JMS STACK2
1127  5724   JMP I ADRPCH
```

```
1130  0000  LTCODE, 0
1131  1100   TAD M200
1132  3346   DCA STK1
1133  1077   TAD P200
1134  4341   JMS PUNCH
1135  2346   ISZ STK1
1136  5334   JMP .-2
1137  7200   CLA
1140  5730   JMP I LTCODE

1141  0000  PUNCH, 0
1142  6046   TLS
1143  6041   TSF
1144  5343   JMP .-1
1145  5741   JMP I PUNCH

1146  0000  STK1, 0
1147  0000  STK2, 0
1150  0177  P177, 177

1151  0000  NUMBIN, 0
1152  4777   JMS DDCB
1153  0020   0020
1154  5751   JMP I NUMBIN

1177  1400
```

```
                /SUBSET COMPILER PAGE 5
                PAGE 5

1200  1054  DIMEN, TAD K2
1201  3032   DCA INCHAR+2
1202  4777   JMS DECOD2            /GET VARIABLE NAME
1203  4104   JMS ERROR             /FOUND A CONSTANT
1204  4776   JMS LISTER            /FOUND VARIABLE , FIX ADDRESS
1205  7201   CLA IAC
1206  1017   TAD 17                /GET VARIABLE NAME POSITION IN LIST
1207  3236   DCA DIM3
1210  4777   JMS DECOD2            /GET DIMENSION
1211  5213   JMP .+2
1212  4104   JMS ERROR             /GOT ANOTHER VARIABLE
1213  7240   STA
1214  1021   TAD 21
1215  7104   CLL RAL
1216  7041   CIA
1217  1067   TAD VARLST            /ADJUST ADDRESS ACCOBDING TO DIMENS.
1220  3067   DCA VARLST
1221  1067   TAD VARLST
1222  3636   DCA I DIM3            /STORE ADDRESS IN TABLE OF VARIABLES
1223  1007   TAD 7
1224  1066   TAD M77
1225  7450   SNA
1226  5775   JMP READ              /END OF RECORD
1227  1774   TAD P5
1230  7450   SNA
1231  5773   JMP CONEND+4          /:
1232  1237   TAD P16
1233  7650   SNA CLA
1234  5202   JMP DIMEN+2           /,
1235  4104   JMS ERROR
1236  0000  DIM3, 0
1237  0016  P16, 16

1240  1054  DATA, TAD K2
1241  3032   DCA INCHAR+2
1242  1071   TAD PRGLST
1243  3323   DCA PRGSAV
1244  4777   JMS DECOD2
1245  4104   JMS ERROR                 /FOUND CONSTANT
1246  4776   JMS LISTER
1247  3324   DCA DATADR
```

```
1250  1007    TAD 7
1251  1772    TAD M50
1252  7450    SNA
1253  5315    JMP DATA2                    /SUBSCRIPTED
1254  1325    TAD M25
1255  7440    SZA
1256  4104    JMS ERROR
1257  4771    JMS NUMBIN
1260  1117    TAD PRSTRT
1261  7041    CIA
1262  1324    TAD DATADR
1263  4770    JMS ADRPCH
1264  1020  DATA3, TAD 20
1265  4767    JMS STACK
1266  1021    TAD 21
1267  4767    JMS STACK
1270  1007    TAD 7
1271  1766    TAD M54
1272  7450    SNA
1273  5244    JMP DATA+4
1274  1327    TAD M21
1275  7650    SNA CLA
1276  5313    JMP DATA4
1277  1323    TAD PRGSAV
1300  4770    JMS ADRPCH
1301  1323    TAD PRGSAV
1302  3071    DCA PRGLST
1303  1007    TAD 7
1304  1126    TAD M72
1305  7450    SNA
1306  5773    JMP CONEND+4
1307  1123    TAD M5
1310  7450    SNA
1311  5775    JMP READ
1312  4104    JMS ERROR
1313  4771  DATA4, JMS NUMBIN
1314  5264    JMP DATA3
1315  4771  DATA2, JMS NUMBIN
1316  7240    STA
1317  1021    TAD 21
1320  7104    CLL RAL
1321  1324    TAD DATADR
1322  5247    JMP DATA+7
1323  0000  PRGSAV, 0
1324  0000  DATADR, 0
1325  7753  M25, -25
1326  7000  NOP
1327  7757  M21, -21
```

```
DDCB=1400

1366  0520  *2000
1367  1075
1370  1124
1371  1151
1372  0565
1373  0252
1374  0525
1375  0222
1376  1014
1377  0600
2000  2000  ENDPROG, .+0
2001  2256   2256                    /R. VARIABLE
2002  7576   7576                    /R. ADDRESS
2003  0000   0000                    /END OF LIST


             /SUBSET MODIFICATIONS TO DIG. 8-29-U-SYM SUBROUTINE

             /SUBROUTINE DICONV - DOUBLE PRECISION INPUT

             /NOW INPUTS FROM BUFFER VIA  TAD I 16 INSTRUCTION

             *1475
1475  7200   CLA
1476  1416   TAD I 16
1477  3007   DCA 7                   /SAVE CHARACTER FOR COMPILER
1500  1007   TAD 7

             *1416
1416  7000   NOP
1417  7000   NOP
1420  7000   NOP

             *1542
1542  7520   -260

             *1544
1544  7740   -40
```

18. OPERATING SYSTEM ASSEMBLY LISTING

```
A       0020
ADDRR   1540
ADDRS   1330
ADRES   0311
ADRESX  0037
ADRESY  0040
ADX     0354
ADY     0355
B       0021
BACK    0224
BACK2   0240
BCDIN   1722
BK2     0074
BNLOAD  0312
BNPRNT  1164
C       0022
CALL    0647
COM     1501
COM1    1337
CRET    0425
D       0023
DADD    1342
DADDS   0353
DAND    0753
DBY2    1703
DHI     1762
DINBAK  1621
DINPUT  1600
DINP2   1601
DIVCNT  1334
DIVGO1  1223
DIVND1  0020
DIVND2  0021
DIVND3  0022
DIVND4  0023
DIV2    1267
DIV3    1245
DLO     1760
DMSK    1756
DMUL    1400
DMUPLY  1167
DPUT    0360
DSAV    1764
DSIN    1757
DSUB    0323
DTEMPX  0356
DTEMPY  0357
DUBDIV  1200
END     0665
```

```
ENDEND 0706
ENDRET 0710
ENDSYS 1777
ERROR  0025
EVAL   0465
EVAL3  0514
EVAL4  0506
EVAL5  0531
EVAL6  0553
EVAL7  0523
EVAL8  0540
FORMAT 0415
GOTO   0625
HDIVSR 1331
HIFORM 0041
IF     0712
IFNEG  0740
IFPOS  0734
IFZER  0736
INDEV  0103
JUMP   0144
KBD    1736
KBD1   1735
KEY    1725
LDIVSR 1332
LIM    0024
LOFORM 0042
LSTNXT 0315
MLTH   1537
MLTL   1536
MP1    1506
MP2    1501
MP3    1566
MP4    1540
MP5    1567
MULTH  1534
MULTL  1535
M1     0051
M100   0575
M11    1155
M11A   1771
M12    1570
M20    1160
M215   1765
M25    1336
M255   1767
M3     1770
M377   1773
M40    0566
M42    0572
M4200  0564
M43    0560
M54    0711
M7     1123
```

```
M70     0073
M700    0033
M7000   0034
M77     0032
M777    0322
NEXT    0313
OP      0047
OPDEV   0044
OPLIST  0107
OPNT    0771
OPRINT  1000
OPRX    0574
OUT     1322
PARAM   0400
PARAMA  0035
PARBK1  0070
PARBK2  0071
PARTEM  0576
PCH     0125
PNTNUM  1103
PNTSGN  1077
PNTZER  1116
POINTE  0573
PRLOAD  0265
PRSTRT  0050
PRST2   0314
PUNCH   0117
PUNCH2  0126
P10     0577
P100    0567
P177    0321
P2      0106
P20     1157
P20A    0072
P212    0562
P212A   1766
P215    0561
P240    0570
P260    1122
P40     1772
P4200   0565
P57     1774
P7      0045
P70     0046
P7600   0036
P77     0571
P7700   0052
READER  1573
READ2   1572
READ3   1713
REM1    1340
REM2    1341
REST    1335
RESU    1533
```

```
RETGTO 0143
SBLOAD 0274
SBNAME 0317
SBSTRT 0316
SDADDR 1124
SDARND 1034
SDBOX  1134
SDCNT  1127
SDCONL 1137
SDDO   1042
SDHIGH 1130
SDHSUB 1132
SDLOOP 1123
SDLOW  1131
SDLSUB 1133
SDMNS  1126
SDOUT  1060
SDPLUS 1125
SDPTR  1136
SDSIGN 1162
SDTEML 1135
SET    0260
SIGNSV 1532
SIGNSW 1333
SKCOUN 1161
SKIPX  1163
START  0202
STFIN  0175
STFIND 0600
STFN2  0605
STFN3  0615
STF2   0104
STF3   0105
SWITCT 0563
TELE   0043
TEMP   0320
TEST7  0744
TEST7A 1156
TEST7B 1755
TEXT0  0432
TEXT1  0444
TLTYPE 0772
TSIGN  1506
TSTERM 0053
TSTER0 0056
TYPE   0763
UNSTAK 0075
```

```
                /SUBSET OPERATING SYSTEM PAGE 0
                 PAGE 0
0000  7000       NOP
0001  4025       JMS ERROR

                *24
0024  0000      LIM, 0
0025  0000      ERROR, 0
0026  7200       CLA
0027  1025       TAD ERROR
0030  7402       HLT
0031  5031       JMP .+0
0032  7701      M77, -77
0033  7100      M700, -700
0034  1000      M7000, -7000
0035  0400      PARAMA, PARAM
0036  7600      P7600, 7600
0037  0000      ADRESX, 0
0040  0000      ADRESY, 0
0041  0000      HIFORM, 0
0042  0010      LOFORM, 10
0043  0772      TELE, TLTYPE
0044  0772      OPDEV, TLTYPE
0045  0007      P7, 7
0046  0070      P70, 70
0047  1000      OP, OPRINT
0050  2000      PRSTRT, ENDSYS+1
0051  7777      M1, -1
0052  7700      P7700, 7700
0053  7000      TSTERM, NOP
0054  4075       JMS UNSTAK
0055  7200       CLA
0056  1007      TSTER0, TAD 7
0057  0046       AND P70
0060  1073       TAD M70
0061  7450       SNA
0062  5474       JMP I BK2
0063  1072       TAD P20A
0064  7650       SNA CLA
0065  5470       JMP I PARBK1
0066  1007       TAD 7
0067  5471       JMP I PARBK2
0070  0401      PARBK1, PARAM+1
0071  0402      PARBK2, PARAM+2
0072  0020      P20A, 20
0073  7710      M70, -70
0074  0240      BK2, BACK2
```

```
0075  0000  UNSTAK, 0
0076  7300   CLA CLL
0077  1416   TAD I 16
0100  3007   DCA 7
0101  1007   TAD 7
0102  5475   JMP I UNSTAK

0103  0000  INDEV, 0
0104  0000  STF2, 0
0105  0000  STF3, 0
0106  0002  P2, 2
0107  0360  OPLIST, DPUT
0110  0025   ERROR
0111  1167   DMUPLY
0112  1342   DADD
0113  0025   ERROR
0114  0323   DSUB
0115  0753   DAND
0116  1200   DUBDIV

0117  2025  PUNCH, 2025   /PUNCH CODE
0120  0000   0
0121  0000   0
0122  1125   TAD PCH
0123  3044   DCA OPDEV
0124  5447   JMP I OP
0125  0126  PCH, .+1

0126  0000  PUNCH2, 0
0127  6021   PSF
0130  5127   JMP .-1
0131  6026   PLS
0132  7300   CLA CLL
0133  5526   JMP I PUNCH2

0134  7000   NOP              /SPARE LOCATIONS
0135  7000   NOP
0136  7000   NOP
0137  7000   NOP
0140  7000   NOP
0141  7000   NOP
0142  7000   NOP
0143  0210  RETGTO, START+6
```

```
                /SUBSET JUMP ROUTINE
                /REFERENCE:-    JUMP S,I,L
                /PROGRAM JUMPS TO STATEMENT (S) IF THE PARAMETER (I) IS
                /LESS THAN OR EQUAL TO THE LIMIT (L)
                /TYPICAL CALL:-    JUMP 37,K=K+2,MAX

0144   1225   JUMP, 1225    /JUMP CODE
0145   0260    SET          /ADDRESS OF NEXT ROUTINE
0146   0000    0
0147   4435    JMS I PARAMA
0150   1023    TAD 23       /GET STATEMENT NUMBER
0151   3146    DCA JUMP+2
0152   4435    JMS I PARAMA
0153   1023    TAD 23       /GET LO-ORDER PARAM. (I)
0154   3105    DCA STF3
0155   1022    TAD 22       /GET HI-ORDER PARAM. (I)
0156   3104    DCA STF2
0157   4435    JMS I PARAMA
0160   1023    TAD 23       /GET LO-ORDER LIMIT
0161   7160    STL CMA      /NEGATE
0162   1105    TAD STF3
0163   7204    CLA RAL
0164   1022    TAD 22
0165   7041    CIA
0166   1104    TAD STF2
0167   7700    SMA CLA      /(I) LESS THAN OR EQUAL TO LIMIT?
0170   5474    JMP I BK2    /NO; GOTO NEXT STATEMENT
0171   1146    TAD JUMP+2   /YES
0172   3023    DCA 23
0173   4575    JMS I STFIN /CHECK STATEMENT NUMBER
0174   5543    JMP I RETGTO /NO SUCH STATEMENT; GOTO NEXT STATEMENT
0175   0600   STFIN, STFIND
```

```
                /SUBSET OPERATING SYSTEM PAGE 1

                PAGE 1

0200   5265     JMP PRLOAD
0201   5274     JMP SBLOAD
0202   6044    START, TPC
0203   6024     PPC
0204   1711     TAD I ADRES
0205   3024     DCA LIM
0206   3715     DCA I LSTNXT
0207   1314     TAD PRST2
0210   3016     DCA 16              /SET PROGRAM START POINT
0211   4075     JMS UNSTAK          /GET INSTRUCTION
0212   1032     TAD M77
0213   7450     SNA
0214   5211     JMP .-3
0215   1033     TAD M700
0216   7650     SNA CLA
0217   5247     JMP BACK2+7
0220   1007     TAD 7
0221   7041     CIA
0222   3320     DCA TEMP
0223   1316     TAD SBSTRT        /GET ADDRESS OF FIRST INSTRUCTION
0224   3317    BACK, DCA SBNAME
0225   1717     TAD I SBNAME
0226   1320     TAD TEMP
0227   2317     ISZ SBNAME
0230   7650     SNA CLA
0231   5236     JMP .+5             /FOUND INSTRUCTION NAME
0232   1717     TAD I SBNAME
0233   7450     SNA
0234   4025     JMS ERROR           /INSTRUCTION LIST EXHAUSTED
0235   5224     JMP BACK
0236   2317     ISZ SBNAME          /FIX INSTRUCTION ADDRESS
0237   4717     JMS I SBNAME        /EXECUTE INSTRUCTION
0240   1007    BACK2, TAD 7         /GET LAST CODE
0241   1032     TAD M77
0242   7450     SNA
0243   5211     JMP START+7         /FOUND END MARK
0244   1033     TAD M700
0245   7440     SZA
0246   5253     JMP .+5
0247   2016     ISZ 16              /FOUND STATEMENT NUMBER; IGNORE
0250   2016     ISZ 16
0251   2016     ISZ 16
0252   5211     JMP START+7
0253   1034     TAD M7000
```

```
0254  7640    SZA CLA
0255  5263    JMP SET+3
0256  7402    HLT                     /FOUND END OF PROGRAM MARK
0257  5206    JMP START+4

0260  2305   SET, 2305                /SET NAME
0261  0625    GOTO                    /ADDRESS OF NEXT INSTRUCTION
0262  0000    0000
0263  4435    JMS I PARAMA            /EVALUATE FOLLOWING PARAMETER
0264  5240    JMP BACK2

0265  6014   PRLOAD, RFC
0266  6032    KCC
0267  3713    DCA I NEXT
0270  1313    TAD NEXT
0271  3315    DCA LSTNXT
0272  1050    TAD PRSTRT
0273  5712    JMP I BNLOAD
0274  6014   SBLOAD, RFC
0275  6032    KCC
0276  1711    TAD I ADRES
0277  1321    TAD P177
0300  0036    AND P7600
0301  3320    DCA TEMP
0302  1320    TAD TEMP
0303  3715    DCA I LSTNXT
0304  1320    TAD TEMP
0305  7001    IAC
0306  3315    DCA LSTNXT
0307  1320    TAD TEMP
0310  5712    JMP I BNLOAD

0311  7753   ADRES, 7753
0312  7715   BNLOAD, 7715
0313  0120   NEXT, PUNCH+1
0314  2002   PRST2, ENDSYS+3
0315  0000   LSTNXT, 0
0316  0144   SBSTRT, JUMP
0317  0000   SBNAME, 0
0320  0000   TEMP, 0
0321  0177   P177, 177
0322  7001   M777, -777
```

```
                /DOUBLE PRECISION SUBTRACT ROUTINE

0323  0000  DSUB, 0
0324  1440   TAD I ADRESY
0325  7161   STL CIA
0326  3357   DCA DTEMPY
0327  7004   RAL
0330  1437   TAD I ADRESX
0331  7041   CIA
0332  3356   DCA DTEMPX
0333  1354   TAD ADX
0334  3037   DCA ADRESX
0335  1355   TAD ADY
0336  3040   DCA ADRESY
0337  4753   JMS I DADDS
0340  5723   JMP I DSUB

                *353
0353  1342  DADDS, DADD
0354  0356  ADX, DTEMPX
0355  0357  ADY, DTEMPY
0356  0000  DTEMPX, 0
0357  0000  DTEMPY, 0

                /DOUBLE PRECISION DEPOSIT ROUTINE

0360  0000  DPUT, 0
0361  1024   TAD LIM
0362  7141   CIA CLL
0363  1037   TAD ADRESX
0364  7620   SNL CLA
0365  4025   JMS ERROR          /DEPOSIT IN PROGRAM AREA
0366  1040   TAD ADRESY
0367  7140   CMA CLL
0370  1036   TAD P7600
0371  7620   SNL CLA
0372  4025   JMS ERROR          /DEPOSIT IN LOADER AREA
0373  1022   TAD 22

0374  3437   DCA I ADRESX
0375  1023   TAD 23
0376  3440   DCA I ADRESY
0377  5760   JMP I DPUT
```

```
                /SUBSET OPERATING SYSTEM PAGE 2

                PAGE 2

0400  0000  PARAM, 0
0401  4075   JMS UNSTAK
0402  1360   TAD M43
0403  7450   SNA
0404  5215   JMP FORMAT              /FOUND # CODE
0405  7001   IAC
0406  7450   SNA
0407  5232   JMP TEXTO               /FOUND " CODE
0410  7001   IAC
0411  7650   SNA CLA
0412  5225   JMP CRET                /FOUND ! CODE
0413  4265   JMS EVAL
0414  5600   JMP I PARAM
0415  4075  FORMAT, JMS UNSTAK
0416  7200   CLA
0417  4265   JMS EVAL
0420  1022   TAD 22
0421  3041   DCA HIFORM              /STORE FORMAT
0422  1023   TAD 23
0423  3042   DCA LOFORM              /STORE FORMAT
0424  5056   JMP TSTERO
0425  1361  CRET, TAD P215
0426  4444   JMS I OPDEV
0427  1362   TAD P212
0430  4444   JMS I OPDEV
0431  5053   JMP TSTERM
0432  3363  TEXTO, DCA SWITCT
0433  4075   JMS UNSTAK
0434  0052   AND P7700
0435  1364   TAD M4200
0436  7450   SNA
0437  5053   JMP TSTERM
0440  1365   TAD P4200
0441  7112   CLL RTR
0442  7012   RTR
0443  7012   RTR
0444  1366  TEXT1, TAD M40
0445  7510   SPA
0446  1367   TAD P100
0447  1370   TAD P240
```

```
0450  4444   JMS I OPDEV
0451  1363   TAD SWITCT
0452  7640   SZA CLA
0453  5232   JMP TEXTO
0454  1007   TAD 7
0455  0371   AND P77
0456  1372   TAD M42
0457  7450   SNA
0460  5053   JMP TSTERM
0461  3363   DCA SWITCT
0462  1007   TAD 7
0463  0371   AND P77
0464  5244   JMP TEXT1

0465  0000  EVAL, 0
0466  3020   DCA 20
0467  3021   DCA 21
0470  3022   DCA 22
0471  3023   DCA 23
0472  7000   NOP
0473  1007   TAD 7
0474  0045   AND P7
0475  1373   TAD POINTE              /FIX ADDRESS OF ARITH. OPERATION
0476  3374   DCA OPRX
0477  1774   TAD I OPRX
0500  3374   DCA OPRX
0501  1007   TAD 7
0502  0052   AND P7700
0503  7440   SZA
0504  5314   JMP EVAL3
0505  4075   JMS UNSTAK              /SIMPLE VARIABLE
0506  3037  EVAL4, DCA ADRESX
0507  1037   TAD ADRESX
0510  7001   IAC
0511  3040   DCA ADRESY
0512  4774   JMS I OPRX              /EXECUTE
0513  5340   JMP EVAL8
0514  1375  EVAL3, TAD M100
0515  7440   SZA
0516  5331   JMP EVAL5
0517  4075   JMS UNSTAK             /VARIABLE SUBSCRIPTED VARIABLE
0520  7001   IAC
0521  3376   DCA PARTEM
0522  1776   TAD I PARTEM
```

```
0523  1051  EVAL7, TAD M1
0524  7104   RAL CLL
0525  3376   DCA PARTEM
0526  4075   JMS UNSTAK
0527  1376   TAD PARTEM
0530  5306   JMP EVAL4
0531  1375  EVAL5, TAD M100
0532  7440   SZA
0533  5353   JMP EVAL6
0534  2016   ISZ 16                    /CONSTANT
0535  1016   TAD 16
0536  2016   ISZ 16
0537  5306   JMP EVAL4
0540  4075  EVAL8, JMS UNSTAK
0541  0046   AND P70
0542  7450   SNA
0543  5273   JMP EVAL+6                /CONTINUE EVALUATION
0544  0377   AND P10
0545  7640   SZA CLA
0546  5665   JMP I EVAL                /FOUND  54,77,777, OR 7777 CODE
0547  7240   STA
0550  1016   TAD 16
0551  3016   DCA 16
0552  5665   JMP I EVAL                /FOUND # OR " OR !
0553  1375  EVAL6, TAD M100
0554  7640   SZA CLA
0555  4025   JMS ERROR                 /UNDEFINED OPERATION
0556  4075   JMS UNSTAK                /CONSTANT SUBSCRIPTED VARIABLE
0557  5323   JMP EVAL7

0560  7735  M43, -43
0561  0215  P215, 215
0562  0212  P212, 212
0563  0000  SWITCT, 0
0564  3600  M4200, -4200
0565  4200  P4200, 4200
0566  7740  M40, -40
0567  0100  P100, 100
0570  0240  P240, 240
0571  0077  P77, 77
0572  7736  M42, -42
0573  0107  POINTE, OPLIST
0574  0000  OPRX, 0
0575  7700  M100, -100
0576  0000  PARTEM, 0
0577  0010  P10, 10
```

```
                PAGE 3

                /DETERMINE ADDRESS OF STATEMENT NUMBER

0600  0000  STFIND, 0
0601  1023   TAD 23                 /GET NUMBER
0602  7041   CIA
0603  3104   DCA STF2
0604  1050   TAD PRSTRT
0605  3105  STFN2, DCA STF3
0606  1505   TAD I STF3
0607  1104   TAD STF2
0610  7640   SZA CLA                /FOUND IT ?
0611  5215   JMP STFN3              /NO
0612  1106   TAD P2                 /YES, DETERMINE ADDRESS OF
0613  1105   TAD STF3               /  FIRST INSTRUCTION
0614  5600   JMP I STFIND
0615  2105  STFN3, ISZ STF3
0616  1505   TAD I STF3             /GET LOCATION OF NEXT NUMBER
0617  7001   IAC
0620  7650   SNA CLA
0621  5474   JMP I BK2              /NO SUCH STATEMENT NUMBER
0622  1505   TAD I STF3
0623  1050    TAD PRSTRT
0624  5205   JMP STFN2

                /SUBSET  GOTO ROUTINE

0625  0717  GOTO, 0717             /GO CODE
0626  0665   END                     /ADDRESS OF NEXT ROUTINE
0627  0000   0
0630  4435   JMS I PARAMA
0631  1007   TAD 7
0632  0072   AND P20A
0633  7650   SNA CLA
0634  5237   JMP .+3
0635  4200   JMS STFIND
0636  5543   JMP I RETGTO
0637  1023   TAD 23
0640  7041   CIA
0641  3104   DCA STF2
0642  4344   JMS TEST7
0643  4435   JMS I PARAMA
0644  2104   ISZ STF2
0645  5242   JMP .-3
0646  5235   JMP GOTO+10
```

```
                /SUBSET CALL ROUTINE

0647  0301  CALL, 0301              /CALL CODE
0650  0712   IF                    /ADDRESS OF NEXT ROUTINE
0651  0000   0
0652  4435   JMS I PARAMA
0653  4200   JMS STFIND
0654  3104   DCA STF2
0655  1007   TAD 7                 /GET PARAMETER TERMINATOR
0656  1311   TAD M54
0657  7700   SMA CLA               /WAS IT # OR " OR !
0660  7240   STA                   /NO
0661  1016   TAD 16
0662  3504   DCA I STF2            /SAVE ADDRESS OF TERMINATOR
0663  1104   TAD STF2
0664  5543   JMP I RETGTO

                /SUBSET END ROUTINE

0665  0516  END, 0516                /END CODE
0666  0647   CALL                /ADDRESS OF NEXT ROUTINE
0667  0000   0
0670  4075   JMS UNSTAK
0671  0046   AND P70
0672  7640   SZA CLA
0673  5306   JMP ENDEND          /UN-NUMBERED END STATEMENT
0674  7240   STA
0675  1016   TAD 16
0676  3016   DCA 16
0677  4435   JMS I PARAMA
0700  4200   JMS STFIND
0701  3104   DCA STF2
0702  1504   TAD I STF2           /GET RETURN ADDRESS
0703  3016   DCA 16
0704  4075   JMS UNSTAK           /GET CALL TERMINATOR
0705  5710   JMP I ENDRET
0706  7402  ENDEND, HLT
0707  5474   JMP I BK2
0710  0241  ENDRET, BACK2+1
0711  7724  M54, -54

                /SUBSET IF ROUTINE

0712  1106  IF, 1106              /IF CODE
0713  0763   TYPE                /ADDRESS OF NEXT ROUTINE
0714  0000   0
0715  4435   JMS I PARAMA            /GET FIRST PARAMETER
0716  1020   TAD 20
0717  7510   SPA                     /POSITIVE ?
```

```
0720  5340   JMP IFNEG                /NO
0721  7440   SZA
0722  5334   JMP IFPOS
0723  1021   TAD 21
0724  7440   SZA
0725  5334   JMP IFPOS
0726  1022   TAD 22
0727  7440   SZA
0730  5334   JMP IFPOS
0731  1023   TAD 23
0732  7650   SNA CLA
0733  5336   JMP IFZER
0734  4344  IFPOS, JMS TEST7
0735  4435   JMS I PARAMA
0736  4344  IFZER, JMS TEST7
0737  4435   JMS I PARAMA
0740  4344  IFNEG, JMS TEST7
0741  4435   JMS I PARAMA
0742  4200   JMS STFIND
0743  5543   JMP I RETGTO
0744  0000   TEST7, 0
0745  7300   CLA CLL
0746  1007   TAD 7
0747  1032   TAD M77
0750  7620   SNL CLA
0751  5744   JMP I TEST7
0752  5474   JMP I BK2
            /DOUBLE PRECISION AND ROUTINE

0753  0000  DAND, 0
0754  1022   TAD 22
0755  0437   AND I ADRESX
0756  3022   DCA 22
0757  1023   TAD 23
0760  0440   AND I ADRESY
0761  3023   DCA 23
0762  5753   JMP I DAND
            /SUBSET TYPE ROUTINE

0763  2431  TYPE, 2431              /TYPE CODE
0764  1725        KEY               /ADDRESS OF NEXT ROUTINE
0765  0000   0
0766  1043   TAD TELE
0767  3044   DCA OPDEV
0770  5771   JMP I OPNT
0771  1000  OPNT, OPRINT
0772  0000  TLTYPE, 0
0773  6041   TSF
0774  5373   JMP .-1
0775  6046   TLS
0776  7300   CLA CLL
0777  5772   JMP I TLTYPE
```

```
                /SUBSET DECIMAL PRINT ROUTINE
                /REFER DIGITAL-8-25-U

                PAGE 4
1000   4435      OPRINT, JMS I PARAMA
1001   1042      TAD LOFORM
1002   7450      SNA
1003   5364      JMP BNPRNT              /FORMAT 0 = PRINT BINARY
1004   1355      TAD M11
1005   3363      DCA SKIPX
1006   1323      TAD M7
1007   3361      DCA SKCOUNT
1010   1022      TAD 22
1011   7700      SMA CLA
1012   1325      TAD SDPLUS
1013   1326      TAD SDMNS
1014   3362      DCA SDSIGN
1015   7100      CLL
1016   1022      TAD 22
1017   7510      SPA
1020   7060      CMA CML
1021   3330      DCA SDHIGH
1022   1023      TAD 23
1023   7430      SZL
1024   7141      CMA CLL IAC
1025   7430      SZL
1026   2330      ISZ SDHIGH
1027   3331      DCA SDLOW
1030   1323      TAD SDLOOP
1031   3327      DCA SDCNT
1032   1324      TAD SDADDR
1033   3336      DCA SDPTR
1034   1736     SDARND, TAD I SDPTR
1035   2336      ISZ SDPTR
1036   3332      DCA SDHSUB
1037   1736      TAD I SDPTR
1040   2336      ISZ SDPTR
1041   3333      DCA SDLSUB
1042   7100     SDDO, CLL
1043   1333      TAD SDLSUB
1044   1331      TAD SDLOW
1045   3335      DCA SDTEML
1046   7004      RAL
1047   1332      TAD SDHSUB
```

```
1050  1330    TAD SDHIGH
1051  7510    SPA
1052  5260    JMP SDOUT
1053  2334    ISZ SDBOX
1054  3330    DCA SDHIGH
1055  1335    TAD SDTEML
1056  3331    DCA SDLOW
1057  5242    JMP SDDO
1060  7200  SDOUT, CLA
1061  1362    TAD SDSIGN
1062  7650    SNA CLA
1063  5303    JMP PNTNUM
1064  1334    TAD SDBOX
1065  7640    SZA CLA
1066  5271    JMP .+3
1067  2361    ISZ SKCOUNT
1070  5316    JMP PNTZER
1071  1362    TAD SDSIGN
1072  1357    TAD P20
1073  7640    SZA CLA
1074  5277    JMP .+3
1075  2363    ISZ SKIPX
1076  5302    JMP .+4
1077  1362  PNTSGN, TAD SDSIGN
1100  1322    TAD P260
1101  4444    JMS I OPDEV
1102  3362    DCA SDSIGN
1103  1334  PNTNUM, TAD SDBOX
1104  1322    TAD P260
1105  4444    JMS I OPDEV
1106  7240    STA
1107  3363    DCA SKIPX
1110  7300    CLA CLL
1111  3334    DCA SDBOX
1112  2327    ISZ SDCNT
1113  5234    JMP SDARND
1114  4756    JMS I TEST7A
1115  5200    JMP OPRINT
1116  2363  PNTZER, ISZ SKIPX
1117  5310    JMP PNTNUM+5
1120  1360    TAD M20
1121  5304    JMP PNTNUM+1
```

```
1122  0260  P260, 260
1123  7771  SDLOOP, -7
1124  1137  SDADDR, SDCONL
1125  7763  SDPLUS, -15
1126  7775  SDMNS, -3
1127  0000  SDCNT, 0
1130  0000  SDHIGH, 0
1131  0000  SDLOW, 0
1132  0000  SDHSUB, 0
1133  0000  SDLSUB, 0
1134  0000  SDBOX, 0
1135  0000  SDTEML, 0
1136  0000  SDPTR, 0
1137  7413  SDCONL, 7413
1140  6700   6700
1141  7747   7747
1142  4540   4540
1143  7775   7775
1144  4360   4360
1145  7777   7777
1146  6030   6030
1147  7777   7777
1150  7634   7634
1151  7777   7777
1152  7766   7766
1153  7777   7777
1154  7777   7777
1155  7767  M11, -11
1156  0744  TEST7A, TEST7
1157  0020  P20, 20
1160  7760  M20, -20
1161  0000  SKCOUNT, 0
1162  0000  SDSIGN, 0
1163  0000  SKIPX, 0
            M7=SDLOOP

1164  1023  BNPRNT, TAD 23
1165  4444   JMS I OPDEV
1166  5314   JMP PNTZER-2

1167  0000  DMUPLY, 0
1170  1037   TAD ADRESX
1171  3374   DCA .+3
1172  4777   JMS I .+5
1173  0022   22
1174  0000   00
1175  3020   DCA A
1176  5767   JMP I DMUPLY
1177  1400    1400
```

```
                /SUBSET OPERATING SYSTEM DIVIDE ROUTINE
                /REFER DIGITAL-8-14-5 DOUBLE PRECISION DIVIDE

                PAGE 5

1200   0000   DUBDIV, 0
1201   1335    TAD REST
1202   3333    DCA SIGNSW
1203   1020    TAD DIVND1
1204   7700    SMA CLA
1205   5223    JMP DIVG01
1206   2333    ISZ SIGNSW
1207   1023    TAD DIVND4
1210   7141    CIA CLL
1211   3023    DCA DIVND4
1212   1022    TAD DIVND3
1213   4737    JMS I COM1
1214   3022    DCA DIVND3
1215   1021    TAD DIVND2
1216   4737    JMS I COM1
1217   3021    DCA DIVND2
1220   1020    TAD DIVND1
1221   4737    JMS I COM1
1222   3020    DCA DIVND1
1223   1437   DIVG01, TAD I ADRESX
1224   7100    CLL
1225   7500    SMA
1226   7060    CMA CML
1227   3331    DCA HDIVSR
1230   1440    TAD I ADRESY
1231   7420    SNL
1232   2333    ISZ SIGNSW
1233   7000    NOP
1234   7430    SZL
1235   7141    CIA CLL
1236   3332    DCA LDIVSR
1237   7430    SZL
1240   2331    ISZ HDIVSR
1241   1336    TAD M25
1242   3334    DCA DIVCNT
1243   7100    CLL
1244   5267    JMP DIV2
1245   1021   DIV3, TAD DIVND2
1246   7004    RAL
1247   3021    DCA DIVND2
```

```
1250  1020    TAD DIVND1
1251  7004    RAL
1252  3020    DCA DIVND1
1253  1021    TAD DIVND2
1254  1332    TAD LDIVSR
1255  3330    DCA ADDRS
1256  7004    RAL
1257  1020    TAD DIVND1
1260  1331    TAD HDIVSR
1261  7420    SNL
1262  5266    JMP DIV2-1
1263  3020    DCA DIVND1
1264  1330    TAD ADDRS
1265  3021    DCA DIVND2
1266  7200    CLA
1267  1023   DIV2, TAD DIVND4
1270  7004    RAL
1271  3023    DCA DIVND4
1272  1022    TAD DIVND3
1273  7004    RAL
1274  3022    DCA DIVND3
1275  2334    ISZ DIVCNT
1276  5245    JMP DIV3
1277  2333    ISZ SIGNSW
1300  5322    JMP OUT
1301  1023    TAD DIVND4
1302  7141    CIA CLL
1303  3023    DCA DIVND4
1304  1022    TAD DIVND3
1305  4737    JMS I COM1
1306  3022    DCA DIVND3
1307  1021    TAD DIVND2
1310  7141    CLL CIA
1311  3741    DCA I REM2
1312  1020    TAD DIVND1
1313  4737    JMS I COM1
1314  3740    DCA I REM1
1315  7240    STA
1316  3020    DCA DIVND1
1317  7240    STA
1320  3021    DCA DIVND2
1321  5600    JMP I DUBDIV
```

```
OUT, TAD DIVND1
1323  3740    DCA I REM1
1324  1021    TAD DIVND2
1325  3741    DCA I REM2
1326  3020    DCA DIVND1
1327  5320    JMP OUT-2
1330  0000   ADDRS,0
1331  0000   HDIVSR, 0
1332  0000   LDIVSR, 0
1333  0000   SIGNSW, 0
1334  0000   DIVCNT, 0
1335  7776   REST, -2
1336  7747   M25, -31
1337  1501   COM1, COM
1340  7576   REM1, 7576
1341  7577   REM2, 7577
             DIVND1=20
             DIVND2=21
             DIVND3=22
             DIVND4=23

             /DOUBLE PRECISION ADD ROUTINE

1342  0000   DADD, 0
1343  7300    CLL CLA
1344  1440    TAD I ADRESY
1345  1023    TAD 23
1346  3023    DCA 23
1347  7004    RAL
1350  1437    TAD I ADRESX
1351  1022    TAD 22
1352  3022    DCA 22
1353  7004    RAL
1354  1021    TAD 21
1355  3021    DCA 21
1356  7004    RAL
1357  1020    TAD 20
1360  3020    DCA 20
1361  1437    TAD I ADRESX
1362  7700    SMA CLA
1363  5742    JMP I DADD
1364  7340    STA CLL
1365  1021    TAD 21
1366  3021    DCA 21
1367  7420    SNL
1370  7240    STA
1371  1020    TAD 20
1372  3020    DCA 20
1373  5742    JMP I DADD
```

PAGE 6

```
                /SUBSET MULTIPLY ROUTINE
                /REFER DIGITAL 8-13-F

1400  0000  DMUL,     0
1401  7300            CLL CLA
1402  1333            TAD RESU            /-2
1403  3332            DCA SIGNSV          /SET SIGN SWITCH
1404  4306            JMS TSIGN           /FETCH AND SET SIGN
1405  1337            TAD MLTH            /RESULT IN MLTH,MLTL
1406  3334            DCA MULTH           /HIGH ORDER MULTIPLICAND
1407  1336            TAD MLTL
1410  3335            DCA MULTL           /LOW ORDER MULTIPLICAND
1411  4306            JMS TSIGN           /FETCH AND SET SIGN
1412  1335            TAD MULTL           /LOW ORDER MULTIPLICAND
1413  3301            DCA MP2
1414  1336            TAD MLTL            /LOW ORDER MULTIPLIER
1415  4340            JMS MP4             /MULTIPLY
1416  3023            DCA D               /LOW ORDER
1417  1367            TAD MP5
1420  3022            DCA C               /HIGH ORDER
1421  1334            TAD MULTH           /HIGH ORDER MULTIPLICAND
1422  3301            DCA MP2
1423  1336            TAD MLTL            /LOW ORDER MULTIPLIER
1424  4340            JMS MP4             /MULTIPLY
1425  1022            TAD C
1426  3022            DCA C
1427  7004            RAL                 /GET CARRY
1430  1367            TAD MP5
1431  3021            DCA B
1432  7004            RAL                 /GET CARRY
1433  3020            DCA A
1434  1335            TAD MULTL           /LOW ORDER MULTIPLICAND
1435  3301            DCA MP2
1436  1337            TAD MLTH
1437  4340            JMS MP4             /MULTIPLY
1440  1022            TAD C
1441  3022            DCA C               /ADD
1442  7004            RAL                 /GET CARRY
1443  1367            TAD MP5
1444  1021            TAD B
1445  3021            DCA B
1446  7004            RAL                 /GET CARRY
1447  1020            TAD A
1450  3020            DCA A               /ADD
1451  1334            TAD MULTH           /HIGH ORDER MULTIPLICAND
1452  3301            DCA MP2
```

```
1453  1337            TAD MLTH          /HIGH ORDER MULTIPLIER
1454  4340            JMS MP4
1455  1021            TAD B
1456  3021            DCA B
1457  7004            RAL
1460  1367            TAD MP5
1461  1020            TAD A
1462  2332            ISZ SIGNSV        /ANSWER <0??
1463  5600            JMP I DMUL        /NO: EXIT
1464  3020            DCA A             /YES
1465  1023            TAD D
1466  7141            CMA CLL IAC       /NEGATE
1467  3023            DCA D
1470  1022            TAD C             /NEGATE
1471  4301            JMS COM
1472  3022            DCA C
1473  1021            TAD B
1474  4301            JMS COM           /NEGATE
1475  3021            DCA B
1476  1020            TAD A
1477  4301            JMS COM
1500  5600            JMP I DMUL        /EXIT
                MP2,
1501  0000      COM,  0
1502  7040            CMA
1503  7430            SZL
1504  7101            CLL IAC
1505  5701            JMP I COM
                MP1,
1506  0000      TSIGN, 0
1507  1600            TAD I DMUL        /FETCH ADDRESS
1510  3340            DCA ADDRR
1511  1740            TAD I ADDRR       /FETCH HIGH ORDER
1512  7100            CLL
1513  7510            SPA               /IS IT <0?
1514  7060            CMA CML           /YES: COMPLEMENT, SET LINK
1515  3337            DCA MLTH
1516  2340            ISZ ADDRR
1517  1740            TAD I ADDRR       /FETCH LOW ORDER
1520  7430            SZL               /WAS IT <0?
1521  2332            ISZ SIGNSV        /YES, ADD 1 TO SWITCH
1522  7000            NOP
1523  7430            SZL
1524  7141            CMA CLL IAC       /COMPLEMENT, CLEAR LINK
1525  3336            DCA MLTL
1526  7430            SZL               /CARRY?
1527  2337            ISZ MLTH          /YES
1530  2200            ISZ DMUL
1531  5706            JMP I TSIGN       /EXIT ROUTINE
```

```
1532  0000  SIGNSV,   0
1533  7776  RESU,     -2
1534  0000  MULTH,    0
1535  0000  MULTL,    0
1536  0000  MLTL,     0
1537  0000  MLTH,     0
            ADDRR,
1540  0000  MP4,      0                    /UNSIGNED MULTIPLY
1541  3306            DCA MP1
1542  3367            DCA MP5
1543  1370            TAD M12              /COUNT 12 BITS
1544  3366            DCA MP3
1545  7100            CLL
1546  1306            TAD MP1              /CARRY GOES INTO
1547  7010            RAR                  /LEFT OF MP1
1550  3306            DCA MP1              /TEST MULTIPLIER BIT
1551  1367            TAD MP5
1552  7420            SNL                  /A 1?
1553  5356            JMP .+3              /NO: DON'T ADD
1554  7100            CLL                  /YES: ADD
1555  1301            TAD MP2
1556  7010            RAR
1557  3367            DCA MP5
1560  2366            ISZ MP3              /DONE 12 BITS?
1561  5346            JMP MP4+6            /NO: CARRY IS IN C(L)
1562  1306            TAD MP1              /YES: DONE
1563  7010            RAR
1564  7100            CLL
1565  5740            JMP I MP4            /EXIT
1566  0000  MP3,      0
1567  0000  MP5,      0
1570  7764  M12,      -14
1571  7000   NOP
1572  1713  READ2, READ3


            /SUBSET INPUT ROUTINES

1573  2205  READER, 2205       /READ CODE
1574  0117   PUNCH
1575  0000   0
1576  1372   TAD .-4
1577  3103   DCA INDEV
1600  4435  DINPUT, JMS I PARAMA
1601  3360  DINP2, DCA DLO
1602  3362   DCA DHI
1603  3357   DCA DSIN
1604  1042   TAD LOFORM
1605  7650   SNA CLA
1606  5322   JMP BCDIN
1607  4503   JMS I INDEV
1610  1367   TAD M255
```

```
1611  7450    SNA
1612  2357    ISZ DSIN              /FOUND  -  SIGN
1613  1370    TAD M3
1614  7510    SPA
1615  5204    JMP DINP2+3
1616  1371    TAD M11A
1617  7740    SLE CLA
1620  5204    JMP DINP2+3
1621  1360  DINBAK, TAD DLO         /FOUND NUMBER
1622  3361    DCA DLO+1
1623  1362    TAD DHI
1624  3363    DCA DHI+1
1625  4303    JMS DBY2
1626  4303    JMS DBY2
1627  1360    TAD DLO
1630  1361    TAD DLO+1
1631  3360    DCA DLO
1632  7004    RAL
1633  1362    TAD DHI
1634  1363    TAD DHI+1
1635  3362    DCA DHI
1636  4303    JMS DBY2
1637  1364    TAD DSAV
1640  0356    AND DMSK
1641  1360    TAD DLO
1642  3360    DCA DLO
1643  7430    SZL
1644  2362    ISZ DHI
1645  4503    JMS I INDEV
1646  1373    TAD M377
1647  7450    SNA
1650  5245    JMP .-3
1651  1372    TAD P40
1652  7450    SNA
1653  5201    JMP DINP2
1654  1374    TAD P57
1655  7510    SPA
1656  5262    JMP .+4
1657  1371    TAD M11A
1660  7750    SGT CLA
1661  5221    JMP DINBAK
1662  7200    CLA
1663  1357    TAD DSIN
1664  7110    CLL RAR
1665  7200    CLA
1666  1362    TAD DHI
1667  7430    SZL
1670  7040    CMA
1671  3437    DCA I ADRESX
1672  1360    TAD DLO
```

```
1673  7430   SZL
1674  7141   CLL CIA
1675  7430   SZL
1676  2437   ISZ I ADRESX
1677  7000   NOP
1700  3440   DCA I ADRESY
1701  4755   JMS I TEST7B
1702  5200   JMP DINPUT
1703  0000  DBY2, 0
1704  1360   TAD DLO
1705  7104   CLL RAL
1706  3360   DCA DLO
1707  1362   TAD DHI
1710  7004   RAL
1711  3362   DCA DHI
1712  5703   JMP I DBY2

1713  0000  READ3, 0
1714  6011   RSF
1715  5314   JMP .-1
1716  6016   RFC RRB
1717  3364   DCA DSAV
1720  1364   TAD DSAV
1721  5713   JMP I READ3

1722  3437  BCDIN, DCA I ADRESX
1723  4503   JMS I INDEV
1724  5300   JMP DBY2-3

1725  1305  KEY, 1305            /KEYBOARD CODE
1726  1573   READER              /ADDRESS OF NEXT ROUTINE
1727  0000   0
1730  1335   TAD KBD1
1731  3103   DCA INDEV
1732  1043   TAD TELE
1733  3044   DCA OPDEV
1734  5200   JMP DINPUT
1735  1736  KBD1, KBD

1736  0000  KBD, 0
1737  6031   KSF
1740  5337   JMP .-1
1741  6036   KRB
1742  3364   DCA DSAV
1743  1364   TAD DSAV
1744  4443   JMS I TELE
1745  1364   TAD DSAV
1746  1365   TAD M215
1747  7640   SZA CLA
1750  5353   JMP .+3
1751  1366   TAD P212A
1752  4443   JMS I TELE
1753  1364   TAD DSAV
1754  5736   JMP I KBD
```

```
1755  0744  TEST7B, TEST7
1756  0017  DMSK, 17
1757  0000  DSIN, 0
1760  0000  DLO, 0
1761  0000   0
1762  0000  DHI, 0
1763  0000   0
1764  0000  DSAV, 0
1765  7563  M215, -215
1766  0212  P212A, 212
1767  7523  M255, -255
1770  7775  M3, -3
1771  7767  M11A, -11
1772  0040  P40, 40
1773  7401  M377, -377
1774  0057  P57, 57
            ENDSYS=1777
            A=20
            B=21
            C=22
            D=23
```

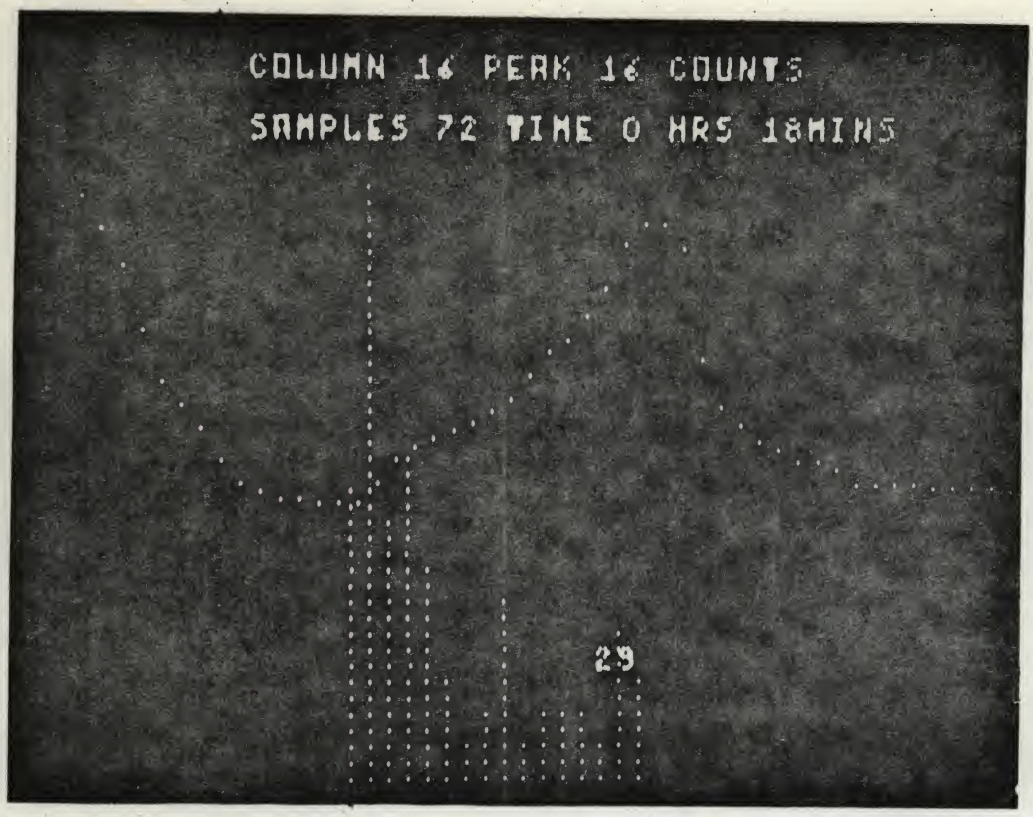


Fig. 1
C. R. T. display.
Last column updated is indicated (29).

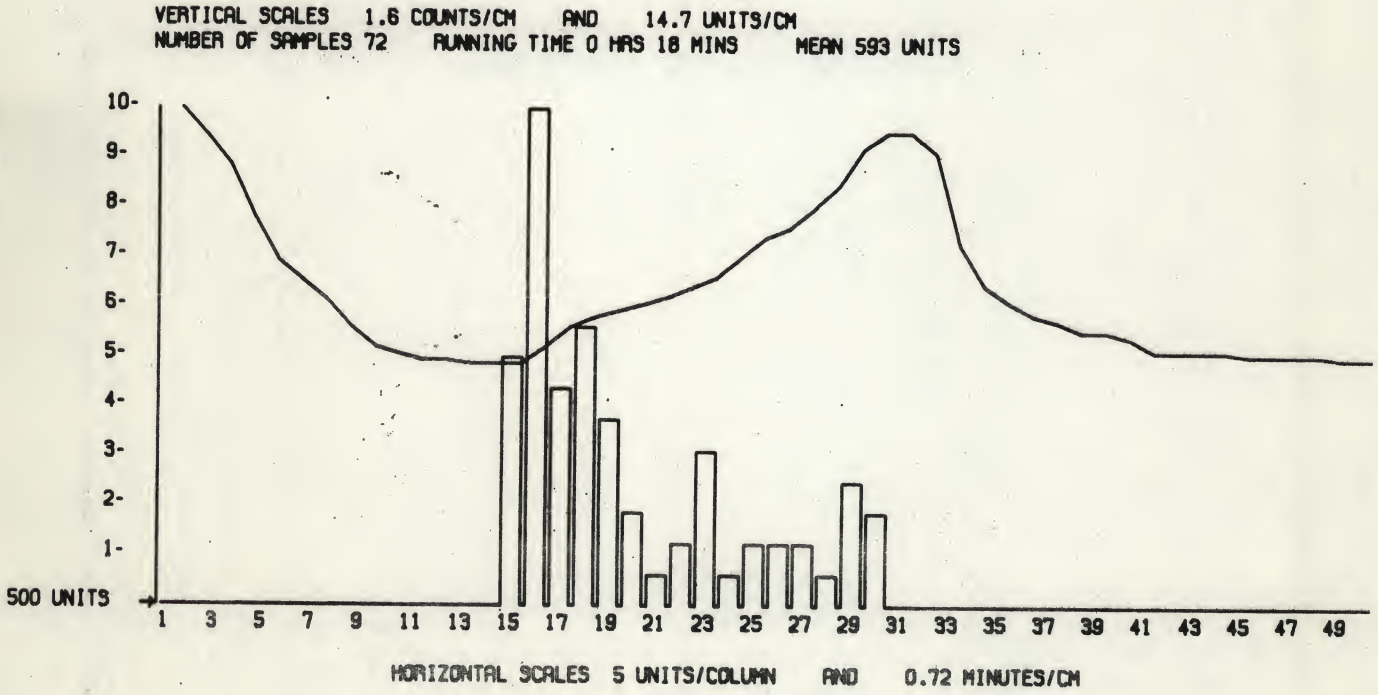


Fig. 2 - Output from incremental plotter

```
        PROGRAM VISILOG
        DATA T.=0,ORIGIN=1,SKIP=2,LINE=3,TEXT=-4,ARROW=-3
        DIMENSION BIN(50),A(50),DISP(50)
        KEY !,"-1 TO START, +1 TO PLOT :",K
        IF K,*START
        CALL *GRAPH
        GOTO *INPUT
*START  KEY !,"CHANNEL ",C," OFFSET ",OFF," UNITS/COLUMN ",S
        SET MAX=1,QMAX=1,K=1,B=0,IT=4,COUNT=1,SAM=0
        SET T.=0,WHEN=4,DV=0,AV=0
   152  SET BIN(K)=0,A(K)=0,DISP(K)=0
        JUMP 152,K=K+1,50
*INPUT  ANALOG C,X
        IF OFF-X,702:SET X=OFF
   702  SET L=X-OFF/S+1,SAM=SAM+1,DV=DV+1,AV=AV+X-OFF
        IF L-51,440:SET L=50
   440  SET BIN(L)=BIN(L)+150,K=1
        TRACE 50,185,#0,0,#1,"COLUMN ",B," PEAK ",MAX," COUNTS"
        TRACE 50,170,"SAMPLES ",SAM," TIME ",T./600," HRS ",R./10,"MI
        SET Z=MAX*150
        IF BIN(L)-Z,507
        SET MAX=BIN(L)/150,B=L
   507  TRACE L*5-6,BIN(L)/MAX,L
        IF T.-WHEN,*SCOPE
        SET TR=T.,A(COUNT)=AV/DV,KK=1,DV=0,AV=0
        IF A(COUNT)-QMAX,1000
        SET QMAX=A(COUNT)
  1000  SET L=KK-1*COUNT/50+1
        SET DISP(KK)=A(L)*150/QMAX
        JUMP 1000,KK=KK+1,50
        JUMP 1004,COUNT=COUNT+1,50
        SET Z=1,QMAX=1,COUNT=26
  1002  SET KK=Z+Z,L=KK-1
        JUMP 1003,A(Z)=A(KK)+A(L)/2,QMAX
        SET QMAX=A(Z)
  1003  JUMP 1002,Z=Z+1,25
        SET IT=IT*2
  1004  SET WHEN=TR+IT
*SCOPE  IF BIN(K),506,505
   506  SET J1=0,POS=K*5,J=BIN(K)/MAX
   600  SCOPE POS,J1=J1+4
        IF J1-J,600
   505  SCOPE K*5+2,DISP(K)
        JUMP *SCOPE,K=K+1,50
        GOTO *INPUT
*GRAPH  SET K=2,TR=T.,Z=DISP
        PLOT ORIGIN,-1000,0,    SKIP,-1018,0,    ARROW,"->"
        PLOT SKIP,-1000,0,      LINE,-1000,1000
        PLOT SKIP,-1300,0,      TEXT,#1,OFF,"UNITS"
        PLOT SKIP,-950,DISP*1000/150
    40  SET Z=DISP(K)+Z
        PLOT LINE,K*50-1000,DISP(K)*1000/150
        JUMP 40,K=K+1,50
        SET K=50
        PLOT SKIP,1500,0
    42  PLOT LINE,X=K*50-1000,0
        PLOT LINE,X,J1=BIN(K)*1000/150/MAX
        PLOT LINE,X-40,J1,      LINE,X-40,0
        IF K=K-1,43,43,42
    43  PLOT SKIP,-1000,1110
        PLOT TEXT,"NUMBER OF SAMPLES ",SAM
        PLOT TEXT,"    RUNNING TIME ",TR/600
        PLOT TEXT," HRS ",R./10," MINS      MEAN "
        PLOT TEXT,Z*QMAX/7500+OFF," UNITS"
        PLOT SKIP,-1000,1160
        PLOT TEXT,"VERTICAL SCALES   ",MAX/10,".",R.
        PLOT TEXT," COUNTS/CM    AND    ",QMAX/10
        PLOT TEXT,".",R.," UNITS/CM"
        PLOT SKIP,-500,-160
        PLOT TEXT,"HORIZONTAL SCALES  ",S
        PLOT TEXT," UNITS/COLUMN    AND    ",TR/250
        PLOT TEXT,".",R./25,R.*2/5," MINUTES/CM"
        SET K=1
    44  PLOT SKIP,K*50-1035,-40,    TEXT,K
        JUMP 44,K=K+2,49
        SET K=0
    45  PLOT SKIP,-110,K=K+100,     TEXT,K/100,"-"
        IF K-1000,45
        PLOT SKIP,-1000,1500
        END *GRAPH
$
```